НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»

УДК ___004.4_____

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Магістерська дисертація

**на здобуття ступеня магістра
за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»
зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «CRM-система танцювального центру»**

Виконав (-ла):
студент (-ка) VI курсу, групи ІТ-93мп
Дмитрук Максим Володимирович _____

Керівник:
Доцент каф. АУТС, к.т.н., доц.
Полторак Вадим Петрович _____

Рецензент:
Завідувач каф. СПіСКС, д.т.н., доц.
Романкевич Віталій Олексійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Дмитруку Максиму Володимировичу

1. Тема дисертації «CRM-система танцювального центру», науковий керівник дисертації Полторак Вадим Петрович, доцент каф.АУТС, к.т.н., доц., затверджені наказом по університету від «26» 10 2020 р. №3132-с

2. Термін подання студентом дисертації _____

3. Об’єкт дослідження: автоматизація процесів системи та доступність даних в електронному вигляді

4. Вихідні дані: мови програмування C# та JavaScript, СУБД MSSQL, технологія ASP.NET Core, технологія Xamarin, бібліотека React, мова розмітки веб-документів HTML, мова стилізації CSS.

5. Перелік завдань, які потрібно розробити: проаналізувати рішення-аналоги, визначення вимог до системи, розробити систему, яка їх задовольняє та описати сценарії її використання.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: діаграма класів та схема бази даних, use-case діаграма, діаграма розгортання системи, діаграми послідовностей, блок схема алгоритму обробки запитів за реалізованим підходом.

7. Орієнтовний перелік публікацій: Winter InfoCom Advanced Solutions 2020.

8. Дата видачі завдання 07.09.2020

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Видача завдання	07.09.2020	
2	Аналіз предметного середовища	09.09.2020 - 21.09.2020	
3	Формування вимог до системи	19.09.2020 – 29.09.2020	
3	Проектування архітектури системи	01.10.2020 – 15.10.2020	
4	Вибір набору технологій для реалізації системи	12.10.2020 - 15.10.2020	
5	Реалізації системи	14.10.2020 – 24.11.2020	
6	Оформлення документації	17.11.2020 – 02.12.2020	
7	Подання роботи до попереднього захисту	02.12.2020	

Студент Максим ДМИТРУК

Науковий керівник Вадим ПОЛТОРАК

АНОТАЦІЯ

Дисертація складається з пояснювальної записки розміром в 109 аркушів. Пояснювальна записка складається з 6 розділів, 8 додатків, 15 посилань на джерела.

Перелік ключових слів: CRM-система, танцювальний центр, автоматизація процесів, доступність даних, підхід, реалізація.

Система створена з метою економії часу усіх учасників процесів в танцювальних центрах за допомогою автоматизації процесів, які відбуваються в ньому, та доступу до необхідної інформації в електронному вигляді. Цілей досягнуто шляхом реалізації CRM-системи танцювального центру, яка складається з двох застосунків.

Кожен з них має клієнт-серверну архітектуру, в якій серверна частина представлена WebAPI застосунком написаною на ASP.NET Core. Для збереження даних використовуються СУБД MSSQL, а взаємодія з нею відбувається через Entity Framework. Один з клієнтських застосунків являє собою веб-застосунок, який реалізовано з використанням фреймворку React. Інший створений для смартфонів та являє собою мобільний застосунок. Його реалізовано з використанням технології Xamarin Forms. Система може бути розгорнута на буд-якій з сучасних операційних систем.

ANNOTATION

The dissertation consists of an explanatory note of 109 sheets. The explanatory note consists of 6 sections, 8 additions and 15 references to sources.

List of keywords: CRM-system, dance center, process automation, data availability, approach, implementation.

The system is designed to save time for all participants in the processes in dance centers by automating those that take place in it, and access to the necessary information in electronic form. The goals were achieved through the implementation of a CRM system of the dance center, which consists of two applications.

Each of them has a client-server architecture, in which the server part is represented by a WebAPI application written in ASP.NET Core. MSSQL DBMS is used for data storage, and interaction with it occurs through Entity Framework. One of the client applications is a web application that is implemented using the React framework. Another is created for smartphones and is a mobile application. It is implemented using Xamarin Forms technology. The system can be deployed on any of the modern operating systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRM - Customer Relationship Management;

ОС – операційна система;

IL – Intermediate Language;

CLR – Common Language Runtime;

ООП – Об’єктно-орієнтоване програмування;

PBKDF2 - Password-Based Key Derivation Function 2;

ПЗ – програмне забезпечення;

СУБД – системи управління базами даних;

SQL – Structured Query Language;

T-SQL – Transact SQL;

DDL – Data Definition Language;

DML – Data Manipulation Language;

DCL – Data Control Language;

IDE – Integrated Development Environment;

ORM – Object-relational Mapping;

CQRS –Command Query Responsibility Segregation;

MVVM – Model-View-ViewModel;

XAML – eXtensible Application Markup Language;

HTTP – Hyper Text Transfer Protocol;

JSON – JavaScript Object Notation;

JWT – JSON Web Token;

HTML – Hypertext Markup Language;

CSS – Cascading Style Sheets;

DOM – Document Object Model;

API – Application Programming Interface;

DI – Dependency Injection;

URL - Uniform Resource Locator;

TPH - Table Per Hierarchy;

RTO - Recovery Time Objective;

RPO - Recovery Point Objective;

AAA - Authentication, Authorization, Accounting;

ВСТУП

В сучасному світі інформаційні технології тісно інтегровані в тотальну більшість сфер життя. Причиною цьому є те, що технології значно прискорюють всі процеси і позитивно відображається на рості бізнесу. Вже сьогодні їхня наявність є обов’язковою, проте недостатньою умовою успішного функціонування компанії. Зовсім скоро бізнеси, які не існують в онлайні – перестануть існувати взагалі.

Не виключенням є сфера розваг, а саме танцювальна індустрія. Тут наявний потік великої кількості клієнтів, дані яких потрібно обробляти та зберігати. Робити це вручну займає досить багато часу та вимагає не малих зусиль. Саме тому використовують автоматизовані системи управління, які прискорюють всі процеси та дозволяють бізнесам працювати більш ефективно. Програмне забезпечення, яке робить можливим швидкий ріст підприємства називається Customer Relationship Management System (CRM-система).

Такий софт економить багато часу працівникам танцювальних центрів, адже робить рутинну роботу набагато швидше, а деяку взагалі автоматизує і не потребує присутності людини. Також технологічний підхід виключає людський фактор, тобто не допускає помилок. Окрім цього, такі системи мають можливість відображати статистику танцювального закладу, яка також допомагає розвиватись бізнесу, відображаючи інформацію необхідну для розвитку центру.

Окрім цього CRM-системи економлять час і клієнтам танцювальних центрів, у яких зникає потреба довго очікувати своєї черги, щоб придбати абонемент або ж дізнатись потрібну інформацію.

Такі системи є досить поширеними в прогресивних танцювальних центрах. Проте існує можливість покращити досвід використання системи танцювального центру. Мова йде про мобільний застосунок для користувачів танцювальних центрів. Такий спосіб

взаємодії з кінцевим користувачем є досить поширеним сьогодні, що може свідчити про його високу ефективність та зручність.

Структура роботи:

В розділі 1 наведено опис предметного середовища та проаналізовано існуючі CRM-системи для танцювальних центрів.

В розділі 2 сформовано вимоги до системи на основі інформації з розділу 1. Їх поділено на функціональні та технічні.

В розділі 3 наведено перелік усіх принципів програмування та технологій, які були використані при створенні системи. Також обґрунтовано доцільність використання кожного з них.

Розділ 4 містить інформацію про реалізацію системи. В ньому наведено процес розробки всіх застосунків, які є в системі та описано як реалізовувались використані підходи.

В розділі 5 міститься інформація про сценарії використання системи з точки зору кожного з видів користувачів.

В розділі 6 описано створення стартап проекту.

1 АНАЛІЗ CRM-СИСТЕМ ДЛЯ ТАНЦЮВАЛЬНИХ ЦЕНТРІВ

1.1 Опис предметного середовища

Сьогодні неможливо уявити будь-яку сферу діяльності без автоматизованих систем управління. Насамперед це аргументується тим, що такі системи оптимізують роботу компанії, пришвидшують її ріст та полегшують роботу клієнтів з нею. Не винятком є і сфера танцювального бізнесу. Тут автоматизовані системи управління економлять багато часу співробітникам шляхом автоматичного формування звітів, їхнього аналізу, тощо. Як відомо, таких систем є достатньо на ринку, тобто вибрати підходящу для себе систему не є складною задачею, проте всі вони є загальними, а не є спеціалізовані для танцювальних центрів. Через це створення спеціалізованої системи для танцювального центру є доцільним.

Зараз в танцювальних центрах використовують CRM-системи загального призначення. Найбільш відомими є Bitrix24, Мегаплан, Microsoft Dynamics 365. Всі перечисленні рішення створювались не для конкретної сфери діяльності, а для обліку даних в цілому. Кожна з цих систем має свої переваги та недоліки, аналіз яких наведено в наступному розділі.

Якщо глянути на ситуацію з іншої точки зору, з позиції клієнта, то автоматизовані системи управління є не настільки важливими для нього. Хоча такі рішення економлять час клієнта, вони не суттєво полегшують його взаємодію з системою. Це стало аргументом для створення застосунку для взаємодії клієнтів з системою.

Основною задачею такого рішення є зробити систему більш доступною та зручною для кінцевих користувачів. Іншими словами, слід надати клієнту можливість отримати бажане в контексті системи без особливих зусиль з його сторони.

Наразі я не зустрічав аналогів таких рішень. Це зумовлено тим, що всі зазвичай намагаються вирішити “біль” власників танцювальних центрів, але ніхто не вирішує

“біль” кінцевих користувачів. А саме це є важливим, оскільки власники танцювальних центрів і намагаються вирішити проблеми клієнтів при взаємодії з їхніми закладами.

Єдиним наявним рішенням для полегшення взаємодії клієнтів з танцювальними центрами є веб-сайт. Більшість можливих взаємодій з системою танцювального центру можна зробити через нього, проте такий спосіб є не завжди зручним. Адже сьогодні більшість дій виконується зі смартфона, а на цьому типі пристроїв зручніше користуватись застосунком, а ніж сайтом. Тому було вирішено створити автоматизовану систему обліку користувачів, однією зі складових якої є мобільний застосунок.

1.2 Огляд існуючих CRM-систем для танцювальних центрів

Для танцювальних центрів використовуються не спеціалізовані CRM, а системи загально призначення. Щоб врахувати переваги, які вони мають, та уникнути недоліків їх використання було проведено огляд найпопулярніших рішень.

1.2.1 Bitrix 24

Це найбільш поширена CRM-система на пострадянському просторі. Це зумовлено тим, що в ній присутній весь необхідний функціонал для ведення бізнесу. За допомогою Bitrix 24 можна контролювати канали комунікації з клієнтами, тобто є можливість поєднати всі соціальні мережі, через які ведеться спілкування з клієнтами, в одному місці. Також ця система дає можливість адмініструвати дані, тобто обмежувати доступ до даних певним користувачам, надати потрібні права доступу користувачеві. Це значно посилює безпеку системи. Її інтерфейс зображено на рисунку 1.1. Цей програмний продукт забезпечений функціональністю контакт-центру. Це особливо зручно, коли всі дії з клієнтом відбуваються всередині однієї системи.

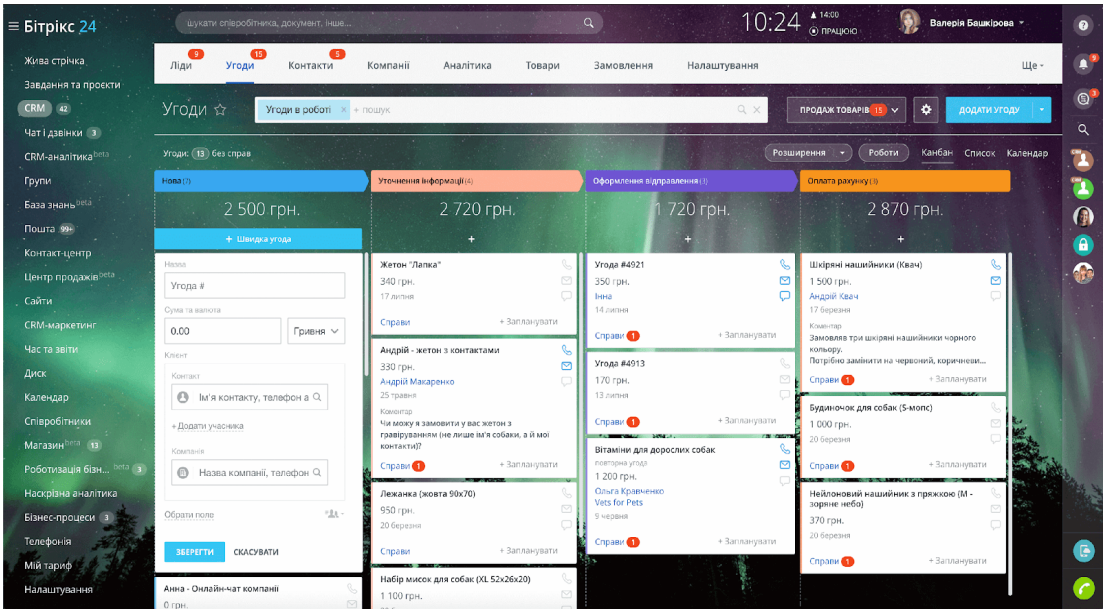

Рисунок 1.1 – Інтерфейс користувача CRM-системи Bitrix 24

Щодо цінової політики Bitrix 24, то тут використовується щомісячна оплата. Як зазначено в [1] найдешевший тариф коштує 3100 грн/міс. А найдорожчий – 10800 грн/міс. При використанні найдешевшим тарифом користувач отримує 10Гб на диску для збереження власних файлів. А при найдорожчому – 1024Гб. В дешевому тарифі доступ до системи отримує 2 адміністратори, а в найбільш дорогому – необмежена кількість користувачів такої ж ролі.

Перевагою Bitrix 24 є популярність цієї системи на пострадянському просторі і велика кількість мають досвід роботи з нею. Також плюсом є багатий набір функціональних можливостей, таких як: відділ продажів, аналітика та звіти, автоматизація продажу та звіти, імпорт/експорт даних, CRM-маркетинг.

З недоліків системи слід виокремити високу ціну за користування. Обґрунтуванням цього недоліку є факт того, що кількість адміністраторів в системі є обмежена і ціна в кожному тарифі є зависокою на ту кількість можливих користувачів такого типу. Також аргументом проти такої високої ціни є обмежена функціональність

системи на дешевих тарифах. І ще одним недоліком є відсутність мобільного застосунку. Доступ до системи тільки через браузер, що не завжди зручно з мобільного пристрою.

1.2.2 Мегаплан

Ця CRM-система є найбільш технологічною і сучасною з усіх проаналізованих аналогів. Окрім звичного для всіх систем такого типу функціоналу, Мегаплан пропонує додаткову функціональність, яка економитиме багато часу співробітникам. Наприклад, в системі є можливість створювати шаблони документів. Це економить час співробітникам та дозволяє їм займатися іншою роботою в цей час.

Також тут є можливість автоматизувати певні задачі, перелік яких наведено далі. Можна створити задачу, яка відправить e-mail на потрібну пошту у вказаній вами годині. Або ж можна налаштувати дію, яка відбуватиметься при настанні вказаних вами умов. Ще система може запропонувати варіанти для заповнення реквізитів користувача, якщо він вже вводив їх раніше. На рисунку 1.2 наведено приклад інтерфейсу системи.

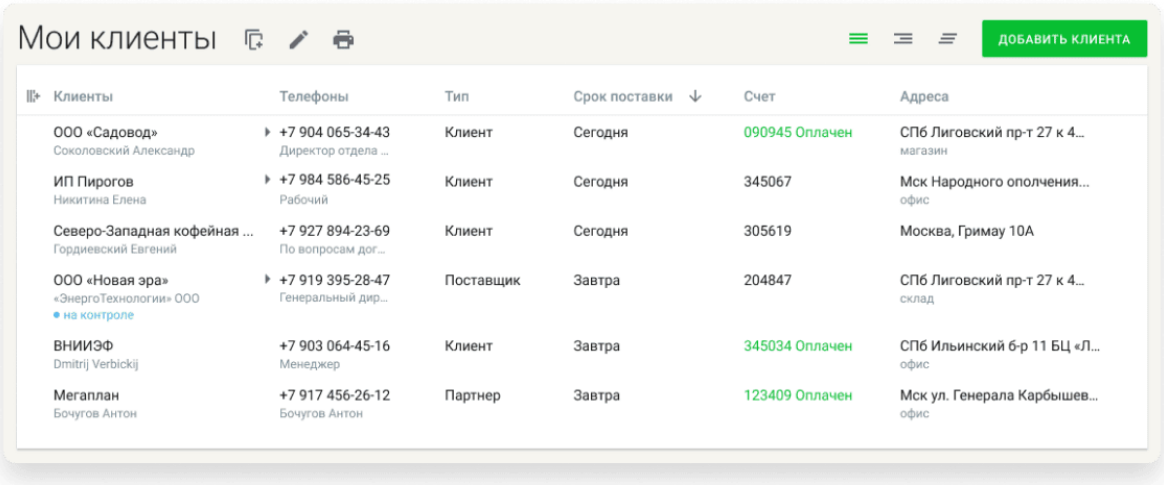

Рисунок 1.2 – Інтерфейс системи "Мегаплан"

Також не можна не відмітити наявність мобільного застосунку в цій системі. Доступність інформації з мобільного пристрою є великим плюсом в сучасному світі.

Цінова політика Мегаплан дає більший вибір своїм клієнтам, оскільки є кілька видів тарифів. Вони визначені в [2]. Перший вид, так звані «коробки», дають можливість користуватись фіксованій кількості користувачів, яка становить 5 адміністраторів. Тут доступно 2 тарифи, які відрізняються лише набором функціоналу. В дешевому тарифі доведеться відмовитись від інтеграції з поштою. Також в більш дешевій версії в системі доступно менше інформації про клієнтів. Окрім цього кількість адміністраторів в системі при використанні найдешевшого пакету послуг є обмежена і становить лише три людини. Ціна дешевшого тарифу – 4110 грн/міс, а дорожчого – 14690 грн/міс.

Також є можливість придбати тариф, де оплата йде за кожного активного користувача в системі. Тобто якщо від вашої компанії системою буде користуватись 5 людей, то ставку слід помножити на 5 і вийде ціна за місяць. Тут ціни варіюються від 126 грн/людину до 290 грн/людину. Слід зазначити що в усіх тарифах адміністраторам доступна необмежена кількість Гб в хмарному сховищі.

Перевагами системи Мегаплан є технологічність та сучасність. Тобто присутня функціональність, така як автоматизація задач або ж можливість створення шаблонів документів. Такого не вистачає в інших системах-аналогах. Також великим плюсом є наявність мобільного застосунку. Без цього дуже складно в сучасному світі. Також великим плюсом є необмежене хмарне сховище, яке доступне навіть в найдешевшому тарифі.

Мінусом системи є висока ціна. Вона помітно більша ніж в аналога Bitrix 24. Ще один недолік Мегаплану полягає в складному інтерфейсі користувача. Посилаючись на офіційний сайт системи, клієнтам потрібно витратити 1-2 тижні на повне освоєння системи. В порівнянні з аналогами це в 2-3 рази більше.

1.2.3 Microsoft Dynamics System

Це CRM-система, розроблена компанією Microsoft, зображена на рисунку 1.3. Вона складається з багатьох ліцензій, що дає можливість обрати більш підходящу для клієнта. Також кожну ліцензію можна налаштувати під свої вимоги і не переплачувати за надлишкову функціональність. Для цього варто обрати необхідний вам набір модулів і тоді у вашій системі буде лише необхідний вам набір функцій. Існують наступні модулі, які можна підключати: фінансовий, маркетинговий, модуль продаж, модуль підтримки користувачів, модуль людських ресурсів. Така можливість робить Microsoft Dynamics System дуже гнучкою у налаштуванні.

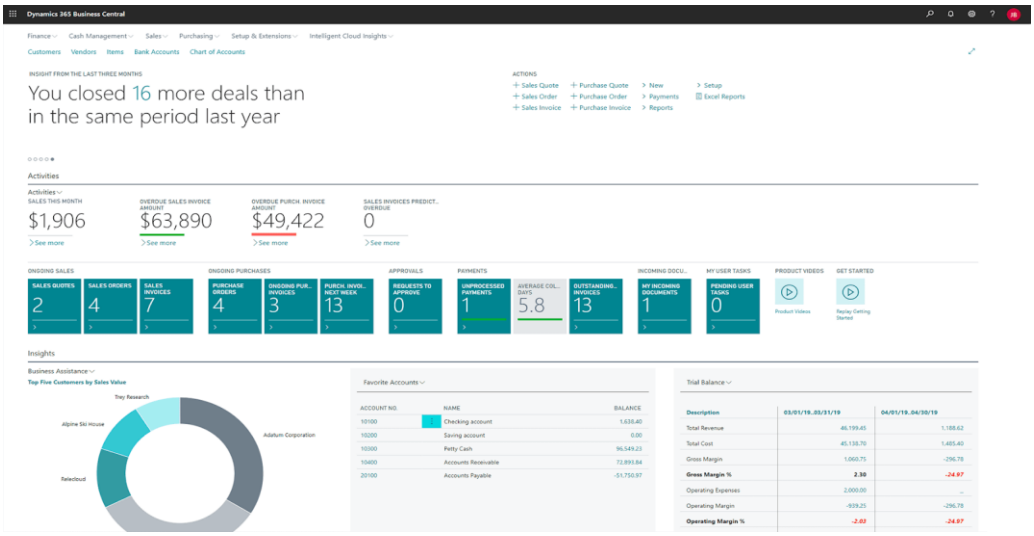

Рисунок 1.3 – інтерфейс користувача CRM-системи Microsoft Dynamics System

Хоча Microsoft Dynamics System є гнучкою в налаштуванні, ціна за модулі є досить висока. В [3] вказано, що ціна за модуль (складову частину ліцензії) може варіюватися від $20 до $1500. За кожного користувача в системі слід платити окремо $8.

Безумовно великою перевагою Microsoft Dynamics System є можливість гнучкого налаштування системи, що дозволяє уникнути наявності надлишкових функцій. Це

економить гроші та час на освоєння системи. На сайті системи зазначено, що час необхідний для навчання використання системи – 1 тиждень. Ще одним плюсом цієї системи є наявність мобільного застосунку. Це є дуже важливим в наш час. З іншої сторони ціна за модулі є дуже висока. Такий недолік знецінює всі переваги системи.

1.2.4 Висновки аналізу оглянутих рішень

На основі аналізу рішень-аналогів сформовано їх порівняння. Його наведено в таблиці 1.1.

Таблиця 1.1 - Порівняння рішень-аналогів

	Bitrix 24	Мегаплан	Microsoft Dynamics System
Ціна	3100-10800 грн/міс	4110-14960 грн/міс	$20-1500/міс
Кількість адміністраторів в системі	2-50	5(ціна нового користувача 290 грн/міс)	безліч(ціна користувача $8/міс)
Об’єм доступного місця для збереження файлів	10	необмежено	10
Переваги	- функціональність; - досвід використання системи великою кількістю людей	- більше корисного функціоналу; - мобільний застосунок; - необмежений обсяг доступного місця	- гнучкість налаштування; - наявність мобільного застосунку

Недоліки	- відсутність мобільного застосунку	- ціна; - складність інтерфейсу користувача	ціна

З порівняльної таблиці можна побачити, що найдешевшим рішенням є Bitrix 24. Незважаючи на обмежену кількість адміністраторів в системі та об’єм доступного місця для збереження файлів в хмарному сховищі CRM, Bitrix 24 є найбільш популярним в нашому регіоні. Тобто важливим критерієм для вибору CRM-системи є її ціна.

1.3 Висновки до розділу

В підрозділі 1.1 наведено опис предметного середовища та вказано причини актуальності системи.

В підрозділі 1.2 міститься аналіз існуючих CRM-систем, які є аналогами створюваної. Для кожного з них визначено переваги та недоліки. Також сформовано таблицю порівняння всіх оглянутих рішень. Найбільш значними перевагами є простота використання системи. З недоліків слід виокремити відсутність важливих функцій, необхідних для танцювальних центрів. Наприклад, в жодної з систем аналогів немає можливості налаштовувати розклад занять. Також жоден з наведених конкурентів не має застосунків для відвідувачів танцювальних центрів.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Проблеми, які вирішує CRM-система

На основі аналізу предметного середовища та всіх переваг й недоліків систем-аналогів можна виокремити проблеми, які покликана вирішити CRM-система танцювального центру. Їх перелік наведено нижче:

- ціна – система повинна бути доступною, вартість має бути конкурентоспроможною на ринку та доступною для її клієнтів;
- зручність використання – в системі повинна бути присутня функціональність, яка по максимуму автоматизує та оптимізує роботу адміністраторів та власників танцювального центру, які працюють з нею;
- зрозумілість – інтерфейс системи повинен бути інтуїтивно зрозумілим, щоб адміністратори не витрачали час на її засвоєння, а клієнти танцювального центру легко могли знайти потрібні їм функції в застосунку;
- мобільний застосунок – повинна бути можливість доступу до системи через мобільний застосунок. Оскільки сьогодні смартфон – це найпопулярніший вид техніки, яким користуються люди, важливим є надати можливість користуватись системою з мобільного пристрою.

З урахуванням цих проблем визначено вимоги до системи. Їх можна поділити на два типи: функціональні та технічні. До функціональних вимог віднесено критерії, важливі з точку зору користувачів системи. Вони вказують на можливості, які повинні бути в системі. До технічних вимог віднесено ті, які необхідні для стабільної та комфортної роботи в системі. Вони стосуються програмного забезпечення, яке входить до складу системи, а не його функціональних можливостей. Два типи вимог спрямовані на покращення роботи системи та полегшення взаємодії з нею кінцевих користувачів. Переліки та пояснення таких вимог знаходиться нижче.

2.2 Функціональні вимоги

Функціональні вимоги та їхні пояснення наведені до застосунку для адміністраторів, власників танцювального центру, викладачів та їхніх відвідувачів є наступними:

- ідентифікація кожного користувача в системі – це необхідно, щоб була можливість відслідковувати історію змін в системі. Тобто буде зрозуміло який користувач зробив ті чи інші дії. Також це дасть змогу керувати доступом до даних, надаючи його лише певній групі користувачів. Це дозволить уникнути ситуації, що працівник одного танцювального центру отримає доступ до функціоналу керування всіма танцювальними центрами, що є в системі;
- адміністрування – налаштування прав користувачів та обмеження прав доступу до певних частин системи. Тобто у власника танцювального центру повинна бути можливість кожній ролі присвоїти перелік функціональності, до якої буде доступ в адміністраторів;
- доступ до потрібної інформації – кожен учасник системи повинен мати доступ до інформації. Наприклад, викладач повинен мати змогу переглянути свій розклад. Або ж клієнт танцювального центру повинен мати доступ до цін танцювального центру;
- віджет для розрахунку вигідного абонемента – віджет, який допоможе розрахувати індивідуально найбільш вигідний абонемент за співвідношенням кількість занять/ціна;
- інтеграція з платіжними системами – для оплати абонементів через мобільний застосунок, слід інтегрувати платіжні системи, які візьмуть на себе обробку транзакцій користувача, роботу з даними кредитних карт;
- сповіщення - нотифікація кінцевого клієнта про актуальні події та майстер-класи в танцювальному центрі. Також сповіщення викладачів та студентів про зміну розкладу, додавання нових уроків, зміну цін на абонементи.

2.3 Технічні вимоги

Технічні вимоги допоможуть вирішити проблеми, зазначені в підпункті 2.1 за допомогою технічних характеристик системи. Ці характеристики наведені нижче:

- надійна – слід виключити можливість втрати даних. У випадку, якщо система перестане функціонувати, дані повинні відновитись з відновленням роботи системи. Також ця вимога означає те, що система не повинна повертати пошкоджені дані, що дасть змогу вберегтись від несумісності даних в майбутньому. RPO(Recovery Point Objective) системи повинен становити 1 рік. Оскільки це максимальна кількість дії абонемента в системі. RTO(Recovery Time Objective) повинен становити 2 дні(48 години). Такий великий RTO зумовлено малою кількістю розробників для зменшенням затрат на команду. Ймовірність відмови системи повинна становити 20% при часі напрацювання системи 12 місяців;
- швидка - очікувана кількість користувачів: 5000 користувачів/танцювальний центр. Всього очікується підключення 3-ьох танцювальних центрів за рік. Тобто система повинна бути розрахована на щорічний приріст користувачів 15000 користувачів. Середній час відповіді сервера повинен бути меншим, ніж 1с;
- безпечна – дані в системі повинні зберігатись в зашифрованому вигляді. В якості алгоритму шифрування паролів використовувати Password-Based Key Derivation Function 2(PBKDF2), оскільки на основі [4] він є найкращим вибором для цієї задачі. Доступ до даних системи повинен бути закритий для неавторизованих користувачів і надаватиметься лише за наявності відповідного ключа доступу,. Для авторизації клієнтів, власників та викладачів танцювального центру в системі використовувати JavaScript Object Notation(JSON) Web Token(JWT). Опис способу авторизації системи та JWT знаходиться в розділі 3;
- зрозумілий інтерфейс - користувач повинен без допомоги інших навчитись користуватись системою менше, ніж за 3 дні;

- сучасна – мобільний застосунок повинен підтримувати найпопулярніші операційні системи: Android та iOS. Мінімальна версія, що підтримується - Android: 7.0 Nougat та iOS 11.

2.4 Висновки до розділу

В підрозділі 2.1 наведено та описано перелік проблем, які покликана вирішити система, що створюється.

Підрозділ 2.2 містить перелік сформованих вимог до системи. Їх поділено на функціональні та технічні. Основними технічними вимогами є швидкість роботи системи та надійність. Її середній час відповіді системи повинен бути менше 1с при очікуваному прирості 15000 користувачів в рік. RTO системи повинен бути 2 дні, а RPO 1 рік. Ймовірність відмови системи при напрацюванні 12 місяців повинна становити 20%.

3 ОБҐРУНТУВАННЯ ВИБРАНОГО НАБОРУ ТЕХНОЛОГІЙ ТА ПРИНЦИПІВ СТВОРЕННЯ WEB ТА МОБІЛЬНИХ ЗАСТОСУНКІВ

3.1 Принципи проектування

Незалежно від складності, наявного функціоналу та призначення застосунку та його складності, існує перелік принципів, які полегшують його розробку та підтримку. Їх слід дотримуватись, щоб в майбутньому можна було легко додати нову функціональність або розширити вже існуючу.

До переліку цих принципів, які були дотримані в магістерській роботі, належать парадигма об’єктно-орієнтованого програмування та принципи програмування S.O.L.I.D.

3.1.1 Об’єктно-орієнтована парадигма програмування

Об’єктно-орієнтована парадигма(ООП) програмування - це принцип мислення розробника та стиль написання коду, в основі якого лежать об’єкти. Кожен об’єкт містить властивості та має певну поведінку. Це визначає інтерфейс взаємодії з ним. Властивості містять в собі корисну інформацію, а поведінка об’єкта дає можливість керувати його властивостями. В ООП існує три основних принципи: інкапсуляція, наслідування та поліморфізм.

Інкапсуляція забезпечує доступ лише до відкритого інтерфейсу взаємодії об’єкта та обмежує до членів класу, які позначені як “private”. Цей принцип знижує ймовірність використати об’єкт не правильно, тим самим уникнути помилок в майбутньому.

Принцип наслідування допомагає уникнути дублювання коду, тим самим пришвидшує процес розробки в рази. Це досягається шляхом побудови ієрархії класів. В класі, який знаходиться вище по ієрархії, визначається спільний функціонал. Цей

функціонал буде доступний в класах, які його наслідують, і його не доведеться дублювати.

Чи не найважливішим принципом ООП є поліморфізм. Наступна фраза допомагає досить влучно його описати: “Один інтерфейс, безліч реалізацій”. Це означає, що цей принцип дає визначити один контракт взаємодії, якого будуть притримуватись усі класи або інтерфейси, які його реалізують, і тоді в ході виконання програми інтерфейс буде підмінюватись на одну з його реалізацій. можливість створити. Це також допомагає зменшити розмір коду шляхом узагальнення логіки. Поліморфізм працює наступним чином, в інтерфейсі описується контракт, а всі класи, що реалізують цей інтерфейс обов’язково повинні дотримуватись цього контракту, реалізувати його. І в майбутньому це дасть змогу використовувати інтерфейс без прив’язки до конкретного класу. Під час виконання програми цей інтерфейс буде змінюватись на одну з його реалізацій. Таким чином поліморфізм робить систему більш гнучкою, пришвидшує її розробку та знижує ймовірність зробити помилку, про що вказано у [5].

3.1.2 SOLID

Принципи SOLID - загальноприйняті практики, основною задачею яких є зменшення ймовірності зробити помилку, зробити код простішим для розуміння і прискорити його написання. SOLID складається з п'яти принципів. Нижче наведено повні назви цих принципів.

- S - Single responsibility principle;
- Open-closed principle;
- L - Liskov substitution principle;
- I - Interface segregation principle
- D - Dependency inversion principle

Single responsibility principle - принцип, зміст якого добре описується наступною фразою: клас повинен мати лише одну причину для змін. Це означає, що клас повинен створюватись з єдиною метою, для виконання лише однієї роботи. Дотримання цього принципу дає кілька переваг. Однією з найбільш значущих є зменшення ймовірності зробити помилку. Адже описувати логіку для однієї задачі легше, ніж для кількох одночасно. Також цей принцип полегшує розуміння коду та структури проекту. Це в свою чергу пришвидшує процес розробки системи, оскільки функціональність в проекті чітко структурована, не перемішана в різних його частинах, тому знайти потрібно логіку в проекті легше.

Open/closed principle - цей принцип свідчить про те, що клас повинен бути відкритий для розширення, але закритий для модифікацій. Тобто, якщо існує необхідність додавання нових можливостей в систему, то це повинно реалізовуватись шляхом доповнення вже існуючої функціональності. Якщо відбудеться зміна вже існуючої функціональності, це вважатиметься порушенням принципу. Цей принцип є важливим, оскільки саме його найчастіше порушують, що призводить до неправильної поведінки системи. Для реалізації цього принципу використовують наслідування та поліморфізм. Нова функціональність прописується в класі-наслідniку. Таким чином стара логіка не змінюється, а нова функціональність додається. Цей принцип допомагає уникати помилок, робить код зрозумілішим.

Liskov Substitution Principle – принцип підстановки Лісков є більш складним для розуміння, ніж попередні принципи. Він полягає в тому, що заміна батьківського типу будь-яким з його класів-наслідників повинна залишати систему працюючою. Цей принцип є дуже актуальним при використанні поліморфізму в системі. Припустимо клас A наслідує клас B. Принцип підстановки Лісков свідчить про те, що при підстановці класу A, замість батьківського класу B в системі не повинно виникнути помилки. Це робить використання поліморфізму більш безпечним, а саму систему більш стабільною. Адже ймовірність виникнення помилки значно зменшується.

Interface Segregation Principle – принцип, який допомагає уникнути надлишкового функціоналу в класах. Його добре описує наступне твердження: жоден клас, що реалізує інтерфейс, не повинен містити функції, які він не використовує. Якщо існує ситуація, коли пара класів реалізують один і той самий інтерфейс і один з цих класів використовує всі методи, а інший містить зайві для нього, то слід інтерфейс розділити на кілька. Це підтримується багатьма мовами програмування, вони дають змогу класам реалізовувати безліч інтерфейсів. Це дає змогу уникнути помилки в майбутньому та зменшити обсяг часу, потрібного для освоєння проекту.

Dependency Inversion Principle – принцип інверсії залежностей. В цьому випадку залежністю називають використання типів користувача в якості параметрів в інших типах користувача. Є кілька трактувань цього принципу. Перше: класи вищого рівня не повинні залежати від класів нижчого рівня, обидва повинні залежати від абстракцій. Друге: абстракції не повинні залежати від деталей, деталі повинні залежати від абстракцій. В цьому контексті абстракції - інтерфейси, які описують лише контракт взаємодії та не містять реалізації. Тобто перше тлумачення означає, що класи, які знаходяться вище по ієрархії, не повинні залежати від класів. Натомість слід використовувати інтерфейси в якості типів параметрів. Друге трактування цього принципу означає, що конкретні класи, які містять реалізацію, повинні використовувати інтерфейси в якості типів параметрів. Це дає змогу на повну використовувати переваги поліморфізму. Дотримання цього принципу допомагає легко дотримуватись open/closed принципу, адже всього лише потрібно використати правильну реалізацію. Також дотримання цього принципу допомагає при написанні юніт-тестів, де потрібно використовувати заглушки, щоб протестувати потрібну функціональність.

Дотримання цих принципів при розробці програмного забезпечення робить код простішим для розуміння і таким чином менше часу необхідно на його опрацювання. Також при дотриманні SOLID систему легше розширювати новим функціоналом.

3.2 Обґрунтування вибору технологій

Система складатиметься з серверного та мобільного застосунків. Для зменшення часу, необхідного для їх реалізації, логічно обрати спільну для цих обох частин мову програмування. Це можливо зробити за допомогою мови програмування - C#. Для розробки веб-застосунку типу Web API можна використати технологію ASP.NET Core. Мобільний застосунок можна реалізовувати за допомогою технології Xamarin. Слід зазначити, що мова C# входить до платформи .NET.

3.2.1 .NET

.NET - платформа для розробників програмного забезпечення, яка була створена компанією Microsoft. Зараз до її складу входить пара фреймворків - .NET Framework та .NET Core. .NET Framework був створений ще на початку цього століття. Він використовується лише для ОС Windows. В той час як другий є кросплатформним і не залежить від ОС. Також .NET Core має відкритий вихідний код, чим також відрізняється від .NET Framework. Це дає змогу розробникам власноруч подивитись як реалізовано певну функціональність і використати оптимальну для їхнього випадку.

Сервери на ОС Windows коштують набагато дорожче, ніж на інших ОС. Це стало найбільш вагомою причиною створення кросплатформеного фреймворка .NET Core. В результаті, такий хід значно збільшив популярність платформи .NET.

За даними ресурсу Stackoverflow, у рейтингу найпопулярніших фреймворків .NET Framework та .NET Core займають друге (35.1%) та третє (26.7%) місця відповідно. В рейтингу найулюбленіших фреймворків .NET Core займає перше місце. 70.7% розробників, які його використовують, зацікавлені в тому, щоб продовжувати розвиватись з його допомогою.

3.2.2 Мова програмування C#

C# - одна з мов програмування доступних в .NET. Вона була створена компанією Microsoft і є найбільш популярною в цій платформі. C# є мовою програмування високого рівня, що означає, що в ній присутні загальновідомі структури даних. Також не потрібно керувати пам'яттю вручну. Це полегшує роботу розробника та економить час при розробці програмного забезпечення.

Окрім цього C# є об’єктно-орієнтованою мовою програмування, що автоматично присвоює їй всі переваги та недоліки цієї парадигми, які були наведені в підрозділі 3.1.1. Вони роблять розробку програмного забезпечення більш гнучкою та швидкою.

В мові програмування C# присутня строга та статична типізація. Строга типізація означає, що тип змінних не можуть бути змінені в ході виконання програми. Хоча це негативно впливає на гнучкість розробки, проте робить систему надійною й дозволяє зекономити час потрібний на її створення. Статична типізація свідчить про те, що перевірка типів відбувається ще на моменті компіляції. Це дозволяє уникнути безлічі помилок в ході виконання програми.

Оскільки мова програмування C# є надійною, дотримується парадигми, яка робить розробку гнучкою та прискорює її, то вона є оптимальним вибором серед існуючих мов програмування. Не можна не згадати, що є досвід використання цієї мови, тому не потрібно буде витрачати час на її освоєння.

3.2.3 Технологія розробки веб-застосунків ASP.NET Core

ASP.NET Core - технологія на платформі .NET для створення веб-застосунків. Вона працює на фреймфорку .NET Core і з’явилась одночасно з ним. Створена вона компанією Microsoft. Ця технологія досить стрімко набирає популярності і за даними Stackoverflow відстає лише на 2% від свого попередника - ASP.NET. Це свідчить про

швидкий розвиток ASP.NET Core і появу нововведень, які пришвидшують розробку програмного забезпечення. Цей факт означає, що інтерес до цієї технології високий і буде легко знайти бажаючих працювати з нею.

Оскільки ця технологія базується на .NET Core, тому вона є кросплатформенною і може працювати на різних ОС. Це є вагомою перевагою над старим ASP.NET, який їй передував, оскільки веб-застосунки розгортають на серверах, які часто працюють на різних ОС. Тепер обмеження по ОС не буде.

Також перевагою ASP.NET Core є відкритий код, який доступний на платформі GitHub. Це дає можливість подивитись як реалізована потрібна функціональність, зрозуміти причини помилок, побачити як можна оптимізувати продуктивність програми.

Також значною відмінністю від попередньої версії технології є доступ до модифікації конвеєра обробки запиту. Це дає можливість додати власний проміжок обробки запиту або прибрати ті, які не потрібні. Тим самим це прискорює роботу системи та робить її більш гнучкою при розробці. Через такі переваги було обрано цю технологію для реалізації серверної частини застосунку.

3.2.4 Технологія розробки кросплатформенних мобільних застосунків Xamarin

Xamarin - технологія для розробки застосунків на мобільні ОС Android та iOS на платформі .NET. Вона була створена компанією Microsoft ще до появи .NET Core. Основною ідеєю технології є кросплатформена розробка, яка дає їй можливість створювати застосунки одночасно на Android та iOS. Xamarin дозволяє уникнути дублювання логіки шляхом винесення її в спільну бібліотеку. Це економить час розробникам та зменшує ймовірність внесення помилки, оскільки за даними Microsoft середньому 90% коду застосунку може використовуватись без змін на різних платформах. По суті Xamarin компілює код написаний на мові програмування C# в

Android-застосунок на Java та в iOS-застосунок на Objective-C, а далі створює мобільні застосунки, як це працює при нативній розробці.

Окрім того, що в Xamarin є можливість створювати логіку застосунку спільно для двох мобільних ОС, також є можливість створювати спільний інтерфейс користувача. Це можна зробити за допомогою технології Xamarin.Forms. Для цього розробнику потрібно описати інтерфейс користувача за допомогою XAML, після чого цей інтерфейс буде доступний на різних ОС з використанням власних елементів керування.

Через можливість написання спільної бузнес-логіки застосунку та інтерфейсу користувача одночасно на дві ОС, швидкість створення програмного забезпечення підвищується, Що є перевагою цієї технології. Xamarin забезпечує простий і зрозумілий інтерфейс користувача, який є однією з вимог до майбутньої системи. Враховуючи ці факти, цю технологію буде використано для створення мобільного застосунку.

3.3 Клієнт-серверна архітектура системи програмного забезпечення

Клієнт-сервер - один з шаблонів архітектури програмного забезпечення для веб-застосунків. Він складається з трьох компонентів: серверної, клієнтської та мережі. Мережа використовується для взаємодії серверної та клієнтської частин. Основна логіка зосереджена в серверній частині системи. Тут визначається вся бізнес-логіка та робота зі сховищем даних. Ця частина системи є централізованою, тобто один серверний застосунок обслуговує багатьох клієнтів. Також на серверній стороні відбувається обробка чутливої інформації(персональних даних, даних про кредитні картки, тощо). Саме тому в цій частині системи багато уваги приділяють безпеці. Клієнтська частина використовується для відображення даних та слугує посередником для взаємодії користувача з системою.

В клієнт-серверної архітектури системи є кілька переваг. Перший плюс полягає в тому, що серверна частина є спільною для всіх клієнтських. У випадку майбутньої CRM-

системи для танцювальних центрів одна серверна частина використовується для мобільних застосунків та для веб-застосунку. Це дасть змогу зекономити час при розробці системи. І оскільки не відбуватиметься дублювання коду зменшиться ймовірність допущення помилки. Також, через відсутність дублювання коду, час на виправлення помилок знижується, оскільки існує лише одне місце, де слід зробити виправлення.

Іншою перевагою клієнт-серверної архітектури є висока продуктивність системи. Це зумовлено тим, що усі ресурсозатратні дії виконуються на серверній частині, а не на кожному клієнтському пристрої. Для серверних частин системи використовують більш потужні комп'ютери, яким потрібно менше часу для вирішення проблем, ніж на клієнтських машинах.

Створюваній системі потрібно взаємодіяти зі сховищем, щоб зберігати там інформацію про користувачів і працювати з нею. З метою підвищення безпеки системи, роботу зі сховищем даних слід вести з серверної частини. Також, аргументом для використання описаної архітектури системи є факт того, що буде кілька клієнтських застосунків. Щоб не дублювати бізнес-логіку та логіку роботи зі сховищем в одному з них, можна зробити це на стороні сервера.

3.4 Система управління базами даних(СУБД) MSSQL, порівняння з аналогами. Технологія об'єктно-реляційного відображення Entity Framework

В системі для керування даними використано СУБД MSSQL. На відміну від аналогів, ця СУБД використовує модифікацію мови Structured Query Language (SQL), яка має назву Transact-SQL. В цій мові існує 3 групи команд:

- команди групи Data Definition Language (DDL) призначені для опису даних при їх створені;

- команди групи Data Manipulation Language (DML) використовуються для додавання, редагування, та видалення даних;
- команди групи Data Control Language (DCL) керують доступом до даних.

До переваг СУБД MSSQL відносять надійність та безпеку даних. Цього досягають шляхом їх шифрування. Також, оскільки ця СУБД, як і використана технологія розробки – ASP.NET Core, створені однією і тією ж компанією - Microsoft, то її використання є досить простим. Також існує документація на офіційному сайті розробника про інтеграцію цих двох технологій.

Також ця СУБД містить кілька особливостей, які відрізняють її від інших. В MSSQL є можливість виконати запит на кілька баз даних одночасно. Це допомагає більш гнучко оперувати даними в системі. Також існує можливість створювати дані, певна інформація яких буде розраховуватись автоматично. Це дозволяє уникнути дублювання коду. Дуже зручною є можливість відмінити запит посеред його виконання. Це економить час, потрібний для написання коду системи та дозволяє уникнути помилок. Оскільки, якби це виконувалось вручну, то будь-яка помилка при виконанні відміни запиту призвела б до розсинхронізації даних, що є критичною проблемою в системі.

В .NET взаємодія з СУБД відбувається за допомогою ADO.NET. Це бібліотека, яка входить до складу платформи. Тобто вона не потребує додаткового встановлення. Хоча ця взаємодія є простою та гнучкою, проте вона потребує багато часу для налаштування. Щоб уникнути цього використовують технологію Object-relational Mapping (ORM). Вона являє собою бібліотеку, яка спрощує роботу з ADO.NET. В платформі .NET існує кілька ORM технологій, найбільш популярною є Entity Framework. Ця бібліотека була розроблена також компанією Microsoft. Оскільки це все продукти однієї компанії, їхня комбінація діє з кращою продуктивністю, ніж інші рішення. І їхнє сумісне використання є більш простим, ніж в інших.

Суть ORM технології полягає у взаємодії з даними зі сховища за допомогою коду. Це дає можливість не писати громіздкі SQL скрипти,а керувати даними з програми. Якщо говорити конкретно про Entity Framework, то існує 2 підходи його використання.

Підхід code first передбачає створення таблиць в базі даних на основі класів, описаних програмним кодом. Інший підхід - database first, діє зворотнім шляхом. На основі створених таблиць в базі даних генеруються класи, які описують сутності. В двох підходах класи використовуються для опису даних, які будуть зберігатись в базі даних та використовуватись в коді.

Далі послідовність дій однакова для всіх підходів. Створюється контекст бази даних. Це клас, в якому визначено таблиці, які містять раніше створені типи даних та конфігурація взаємодії з базою даних. Тут або в файлі конфігурації проекту потрібно вказати рядок підключення до бази даних. Після генерується файл, який описує зміни, які буде застосовано до бази даних після його виконання. Він називається міграцією. Далі слід виконати команди в терміналі, які застосують зміни до бази даних. Таким чином зміни в коді переносяться на базу даних.

Опис СУБД MSSQL свідчить про те, що вона має потрібні характеристики, тому можу бути використана в розроблюваній системі. А її просто використання з іншими продуктами компанії Microsoft робить її оптимальним варіантом.

Аргументи за використання технології Entity Framework - спрощення та прискорення розробки. Спрощення буде через те, що є досвід використання цієї технології. А прискорення розробки буде тому, що з’явиться можливість використовувати всі переваги, які є в технологіях платформи .NET.

3.5 Опис підходу Command Query Responsibility Segregation(CQRS)

В серверній частині системи буде використано підхід CQRS. Його зміст полягає в розмежуванні команд та запитів. В контексті цього підходу командою називають будь-

який запит, який змінює стан системи і не повертає ніякого значення. Наприклад, створення заняття, його видалення або редагування. Натомість запит, в контексті підходу CQRS не змінює стан системи, а лише повертає дані. Принцип за яким працює система з цим підходом описано в [6; 7] та на рисунку 3.1

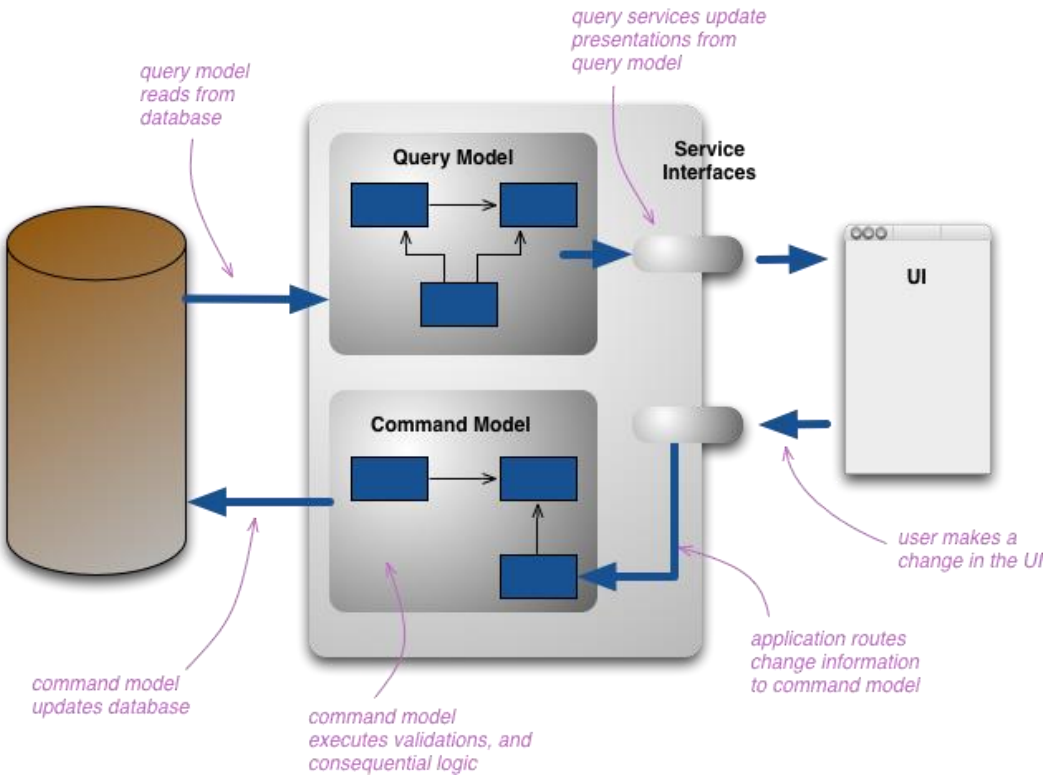

Рисунок 3.1 – Принцип роботи підходу CQRS

В системі цей підхід реалізовано наступним чином. Користувач з клієнтського застосунку виконує потрібні йому дії, в результаті чого поступає http запит на серверну частину. Там він потрапляє в контролер, де з нього формуємо команду або запит. Після передаємо сформовану команду/запит на виконання в Bus, в якому відбувається валідація отриманих даних та виклик потрібного обробника. І якщо це був запит, то у відповідь отримуємо результат його виконання, яким оперуємо далі.

Такий підхід робить систему більш передбачуваною. Оскільки якщо виконується якийсь запит, то можна точно сказати, що він не змінює стану системи. І це працює для

команд, вони не повертають дані, а змінюють визначений стан системи, що є очікуваним. Як результат, це допомагає знизити ймовірність допущення помилок, робить проект більш зрозумілим та легким для початку роботи з ним.

У такого підходу є значний недолік - потрібно витратити час, щоб організувати проект потрібним чином. В середньому на створення необхідних інтерфейсів та їх реалізацію потрібно близько 4 годин. Тому цей підхід не варто використовувати в проектах малого розміру. В такому випадку більше часу витратиться на організацію проекту, ніж на написання самої бізнес-логіки. Оскільки в розроблювана система займе більше часу для реалізації, то дотримання такого підходу є доцільним і принесе лише покращення в проект. Програмний код стане зрозумілішим, а це зробить проект легким для підтримки в майбутньому.

3.6 Опис паттерна Model-View-ViewModel(MVVM)

Паттерн MVVM буде використано в клієнтській частині системи - в мобільному застосунку. Основною ідеєю цього паттерну є розмежування логіки застосунку від інтерфейсу користувача. В цьому контексті логіка застосунку - все, що відбувається на клієнтській частині і є приховане від погляду користувача. До такої логіки відноситься комунікація з серверною частиною системи, валідація даних та певна бізнес-логіка, яка не сильно навантажує систему, наприклад, розрахунок оптимального абонементу. Принцип за яким працює мобільний застосунок описано в [8] та на рисунку 3.2.

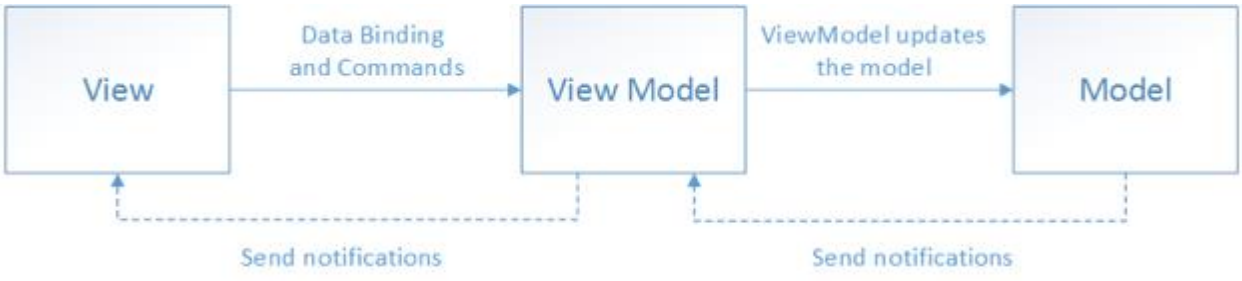

Рисунок 3.2 – Принцип роботи паттерна MVVM

В системі цей підхід буде реалізовано в проектах для мобільних пристроїв на Xamarin. Кожна сторінка, яку бачитиме користувач, буде описана за допомогою XAML і знаходитиметься в папці Pages. Це слугує View в створеній системі. Коли користувач натисне кнопку або перейде на сторінку, відбудеться виклик проміжкового шару, який служитиме ViewModel. ViewModel для кожної сторінки буде різна. В ній будуть знаходитись команди та властивості. Властивості зв'язані з елементами інтерфейсу користувача. Коли значення в цьому елементі буде змінюватись, зміниться і значення у відповідної властивості в ViewModel. Команди будуть викликатись при натисканні кнопок на View. В результаті виклику команди запит буде переходити в відповідний сервіс, який виконуватиме обов'язки Model.

За допомогою паттерна MVVM код інтерфейса користувача залишатиметься відокремленим від іншої логіки в мобільному застосунку. Це зробить принцип роботи системи простішим для розуміння, що в прискорить розробку застосунку або полегшить знайти помилку, оскільки буде зрозуміло в якій частині вона виникла.

Цей патерн є єдиним для розробки мобільних застосунків на Xamarin. Патерни, які використовуються при розробці на інших мовах програмування, є дуже схожими за змістом і відрізняються лише назвою. А оскільки MVVM робить розробку мобільного застосунку швидшою, а його програмний код легшим для розуміння, це є аргументами для його використання в майбутній системі.

3.7 Стандарт авторизації OAuth 2.0 та JWT

Для безпечної авторизації користувачів в системі сьогодні використовують стандарт OAuth 2.0. По своїй суті цей стандарт являє собою перелік алгоритмів авторизації для різних типів систем. За [9] розрізняють 3 алгоритми авторизації:

- по авторизаційному коду - використовується для авторизованої комунікації між виключно серверними застосунками;

- неявний - використовується в мобільних та веб-застосунках з використанням сторонніх сервісів авторизації користувачів. Відомими є авторизація за допомогою Google та Facebook;
- за даними клієнта - використовується в багатьох десктопних, мобільних та веб-застосунках. Авторизація відбувається за допомогою логіну та паролю.

В системі використано алгоритм авторизації за даними клієнта. Такий вибір зумовлено легкістю його реалізації. Він діє наступним чином. На клієнтській частини користувач вводить свої дані для авторизації. Після чого ці дані передаються на серверну частину. Там відбувається перевірка даних.

Паролі користувачів в системі зберігаються в зашифрованому вигляді. Як зазначено в [10;11;12], це зроблено з метою підвищення безпеки даних користувачів. У разі їх викрадення, не буде можливості їх скомпрометувати. Тому, щоб перевірити чи вірні дані ввів користувач, слід зашифрувати пароль і таким чином шукати запис в базі даних. Для шифрування обрано один з доступних способів -

У випадку якщо дані не вірні, то повертається помилка і користувач залишається не авторизованим. Якщо отримані дані правильні, то формується access token. В ньому містяться дані, необхідні для ідентифікації користувача при майбутніх запитах. Після сформований access token повертається на клієнтську частину і зберігається там. Всі запити, які потребують авторизації користувача, виконуються з access token’ом. Його передають в заголовку запиту.

Для access token’а використано стандарт JWT. Посилаючись на [13] він собою являє зашифрований JSON об’єкт, який складається з трьох частин розділених крапками: заголовок, корисна частина та підпис. Цей стандарт є легким для розуміння, використання та реалізації, за що його і було обрано. Для його використання існують бібліотеки для різних мов програмування, які мають весь необхідний функціонал щоб працювати з ним, а саме виконують формування та валідацію токенів.

Описаний стандарт OAuth 2.0 та технологія JWT забезпечують надійність даних при передачі. З їх допомогою буде реалізовано протокол AAA(Authentication, Authorization, Accounting) в створюваній системі.

3.8 Веб-технології HTML, CSS, препроцессор SASS

Візуальна частина клієнтського веб-застосунку створена з використанням мови розмітки Hypertext Markup Language (HTML) та мови стилів Cascading Style Sheets (CSS).

Мова розмітки HTML використовується, щоб формувати каркас веб-сторінок. Вона є найбільш популярною мовою за своїм призначенням. На відміну від аналогів, HTML підтримується всіма браузерами та має багато документації. Вона складається з елементів. Кожен елемент містить в собі початковий тег, контент та закриваючий тег. Якщо елемент не містить контенту, його представляють у лише вигляді тегу. Тег складається з назви та може містити атрибути.

Мова стилів CSS описує спосіб відображення елементів створених в розмітці сторінки. Вона стандартизована специфікацією W3C, що свідчить про те, що ця мова достатньо документована. CSS є найбільш популярною мовою стилів. З її допомогою можна визначити стилі для конкретних елементів. Для ідентифікації елемента можна використовувати назву тега, його клас або ж ідентифікатор. Хорошою практикою вважається використовувати саме класи, адже це є їхнє основне призначення.

Щоб прискорити процес стилізації сторінки використовують препроцесори. Вони допомагають уникнути дублювання коду, тому що надають можливість визначати змінні і повторно використовувати їх в різних файлах та дозволяють вказувати стилі вкладеним елементам.

Відрізняються препроцесори функціями, які містять в собі. Наприклад, є можливість застосовувати стилі лише за настання певних умов або можливість

визначати стилі для елементів з використанням циклів. Зважаючи на [14] найбільш функціональним є препроцессор SASS. Тому для надання додаткової гнучкості при стилізації веб-сторінок його буде обрано для використання в майбутній системі

3.9 Мова програмування JavaScript та фреймфорк React

Щоб окрім гарного вигляду веб-сторінок, надати можливість виконувати певну функціональності використовують мову програмування JavaScript. Для цієї мови програмування існує стандарт ECMAScript, в якому визначено всі правила. А JavaScript повністю реалізує його.

JavaScript мова програмування з слабкою та динамічною типізаціями. Це означає, що перевірка типів відбувається в процесі виконання програми, а змінні, залежно від значення, яке їм присвоюють, можуть змінювати свій тип.

Для взаємодії з веб-сторінкою використовують Document Object Model (DOM). DOM являє собою деревоподібну структуру, точками розгалуження якого є елементи, визначені в розмітці сторінки. Є визначено Application Programming Interface(API) взаємодії з DOM сторінки. Але існує вагомий недолік його використання - він потребує написання великої кількості коду. Для уникнення цього створено кілька фреймворків, які інкапсулюють логіку роботи з DOM та визначають свій, більш простий для використання API взаємодії та життєвий цикл веб-сторінки.

З доступних веб-фреймворків найбільш популярним є React. Він був створений компанією Facebook для внутрішнього використання, але з часом його зробили відкритим. В ньому всі веб-сторінки складаються з компонентів. Кожен компонент має свій стан та властивості. Стан змінний, а властивості - ні, вони приходять з батьківського компоненту(той, який викликав створення цього компоненту). Кожен компонент містить метод render, який повертає модифікацію мови розмітки Extensible Markup Language (XML). Така модифікація називається JavaScript XML(JSX). Після компіляції

вона перетворюється в звичайний HTML, який відображається на веб-сторінці. React бере на себе обробку зміни стану системи і оптимізує звернення до DOM веб-сторінки. Цим він полегшує роботу, оскільки дозволяє уникнути помилок та зекономити час розробки.

Оскільки React помітно оптимізує роботу з DOM веб-сторінки, та за [15] оптимізує розширення системи, яка його використовує, є легшим, ніж в аналогів, в системі буде використано саме його.

3.10 Висновки до розділу

В підрозділі 3.1 наведено переваги та недоліки, які надають ООП та принципи SOLID при розробці програмного забезпечення. Також обгрунтоване рішення дотримуватись цієї парадигми та принципів в майбутній системі.

В підрозділі 3.2 обґрунтовано обрані технології: платформу .NET в цілому, фреймворк .NET Core та технології для розробки веб-застосунків ASP.NET Core і для мобільних застосунків - Xamarin. Перераховано, які переваги актуальні при розробці створюваної системи та як вони зможуть забезпечити визначені вимоги до системи.

В підрозділі 3.3 описано клієнт-серверну архітектуру програмного забезпечення. Вказано принцип її роботи та яким чином відбувається взаємодія двох компонентів в системі.

Підрозділ 3.4 містить опис СУБД MSSQL. Визначено переваги, які дає ця система та проаналізовано її використання в контексті сиситеми, яка розроблятиметься. Також в цьому підрозділі описано ORM технологію Entity Framework та як вона вплине на систему.

В підрозділі 3.5 описано підхід CQRS, який планується використовувати в серверній частині. Пояснено його зміст, переваги, які він принесе в систему та розказано про спосіб його реалізації в системі.

Підрозділ 3.6 містить опис паттерна MVVM для мобільних застосунків. Наведено спосіб реалізації цього патерну в системі та пояснено принцип його роботи. Також вказано його переваги та вигоду його використання в майбутній системі.

Стандарт авторизації OAuth 2.0 пояснено в підрозділі 3.7. Описано принцип його роботи в загальному та конкретно в розроблюваній системі. Наведено вимоги, які забезпечує його дотримання. Також міститься інформація про JWT, який буде використаний для формування access token`a.

В останніх двох підрозділах, 3.8 та 3.9, описано технології, які будуть використовуватись для створення клієнтських веб-застосунків. Описано суть мови розмітки HTML та мови стилів CSS. Також наведено переваги мови програмування JavaScript для використання на клієнтській частині системи та обґрунтовано доцільність використання веб-фреймворку React.

4 РОЗРОБКА CRM-СИСТЕМИ ТАНЦЮВАЛЬНОГО ЦЕНТРУ

Створена система складається з п'яти компонентів: бази даних, серверної та клієнтської частин застосунку DOM CRM та серверної і клієнтської частин мобільного застосунку DOM. Візуально діаграму компонентів можна побачити в додатку А. Там описано фізичні машини, на яких працює кожен з компонентів системи, середовища розгортання та способи комунікації між ними.

4.1 Розробка клієнт-серверної архітектури

В системі присутні два застосунки. Кожен з них реалізовано з використанням клієнт-серверної архітектури. Такий підхід передбачає, що застосунок складається з двох частин - серверної та клієнтської.

Серверна частина виконується на сервері, який відзначається більшою потужністю, ніж пристрої користувачів. Це зумовлено тим, що саме в цій частині клієнт-серверної архітектури виконуються ресурсозатратні операції. Також серверна частина слугує для взаємодії з базою даних. Ще в цій частині прописана бізнес-логіка застосунку. Це зроблено з метою підвищення безпеки системи, адже доступ до серверної частини мають лише її розробники. Також рішення зосередити тут функціонал, який займає багато часу, покращує продуктивність системи. Для розуміння слід враховувати, що один серверний застосунок використовується багатьма клієнтськими. Тобто різні користувачі, користуючись застосунком окремо один від одного, все рівно виконують запит, який потрапляє на один і той самий сервер. Важливо це пам’ятати при розробці цієї частини системи.

Клієнтська частина буде працювати на машинах кінцевого користувача застосунку. Це може бути комп’ютер, ноутбук, планшет або смартфон, в залежності від виду клієнтського застосунку. В ній знаходиться логіка валідації даних, які ввів

користувач. Така валідація не є складною і полягає лише в перевірці формату введених даних. Також в клієнтських застосунках присутня логіка, яка надає інтерактивності інтерфейсу користувача. Наприклад, при натисканні кнопки “Login” спрацьовує обробник цієї події, який виконує певні дії. Ще в клієнтській частині знаходиться логіка для взаємодії з серверною частиною системи. Адже щоб передати отримані від користувача дані, потрібно їх надіслати на сервер. А по отриманні відповіді від сервера, клієнтський застосунок обробляє його та відображає його в зручному для користувача форматі.

В створеній системі в обох застосунках серверна частина реалізована з використанням технології ASP.NET Core. Причини вибору саме її можна побачити в підрозділі 3.2.3. Робота такого виду застосунку починається з файла Program.cs, в якому налаштовується сервер Kestrel. В реалізованій системі не використано додаткового налаштування сервера, оскільки конфігурація за замовчуванням є підходящою. Тут вказано використовувати файл Startup.cs для налаштування застосунку. В ньому присутні два методи. Перший, ConfigureServices, призначений для реєстрації залежностей. В застосунках ASP.NET Core присутній спосіб dependecny injection за замовчуванням. Він полягає в тому, що в методі ConfigureServices в якості параметра передається колекція сервісів, в яку потрібно додати всі залежності, які потім будуть використовуватись в застосунку. Другий метод, який присутній в класі Startup - метод Configure. Він використовується для визначення конвеєра обробки запиту. Тут визначається як буде оброблятися кожен запит в застосунку. Вказується яким чином вирішувати на який endpoint піде запит в залежності від шляху. Також є можливість додати свій обробник, який буде спрацьовувати при кожному запиті або лише при старті застосунку. В системі присутній самописний обробник, рисунок 4.1, який застосовує міграцію на базу даних, якщо є нові.

```
public static IApplicationBuilder UseAutoMigration(this IApplicationBuilder builder)
{
    var services = builder.ApplicationServices;

    using var serviceScope = services.CreateScope();

    using var domDbContext = (DOMDbContext) serviceScope.ServiceProvider.GetService(typeof(IDOMDbContext));

    domDbContext.Database.Migrate();

    return builder;
}
```

Рисунок 4.1 – Middleware для застосування міграцій до бази даних

Після реєстрації сервісів та визначення конвеєра обробки запитів, застосунок готовий до роботи. Він приймає запити, пропускає їх через всі стадії обробки. На одній зі стадій запит потрапляє в контролер. Контролер являє собою клас, який наслідується від класу ControllerBase. Таким чином він переймає від батьківського класу функціонал, який часто використовується при обробці запитів. Таким чином він це пришвидшує розробку.

Кожен контролер містить методи, в які і потрапляє запит. Ці методи називається кінцевими точками (endpoint) взаємодії. В них починається процес обробки запиту. Кожна точка вхаємодії має тип запиту, який вона може обробити. Цей тип визначається за допомогою атрибуту та вказується до сигнатури метода.

В прикладі, зображеному на рисунку 4.2, виконується обробка запитів типу Patch. Про це свідчить атрибут HttpPatch. Також існують інші атрибути, які відповідають типам запитів: HttpGet, HttpPost, HttpDelete, HttpPut, HttpPatch. В прикладі атрибут містить параметр, який використовується на стадії роутингу, коли вирішується в яку кінцеву точку передати цей запит.

```
[HttpPatch("phone-number")]
public Task UpdatePhoneNumber([FromBody] UpdateStudentPhoneNumberCommand command)
{
    return Execute(command);
}
```

Рисунок 4.2 – Точка взаємодії системи

Для зручності роботи зі створеною серверною частиною її описано за допомогою автоматично згенерованої документації за допомогою бібліотеки Swagger. Принцип її роботи полягає в аналізі доступного проекту та знаходженні всіх контролерів. Після цього в кожному з контролерів визначається кінцеві точки взаємодії, їхні вхідні параметри та моделі відповідей. Також враховуються типи запитів, які відповідають кожній з точок. Приклад генерації документу, який описує CRM DOM API наведено на рисунку 4.3.

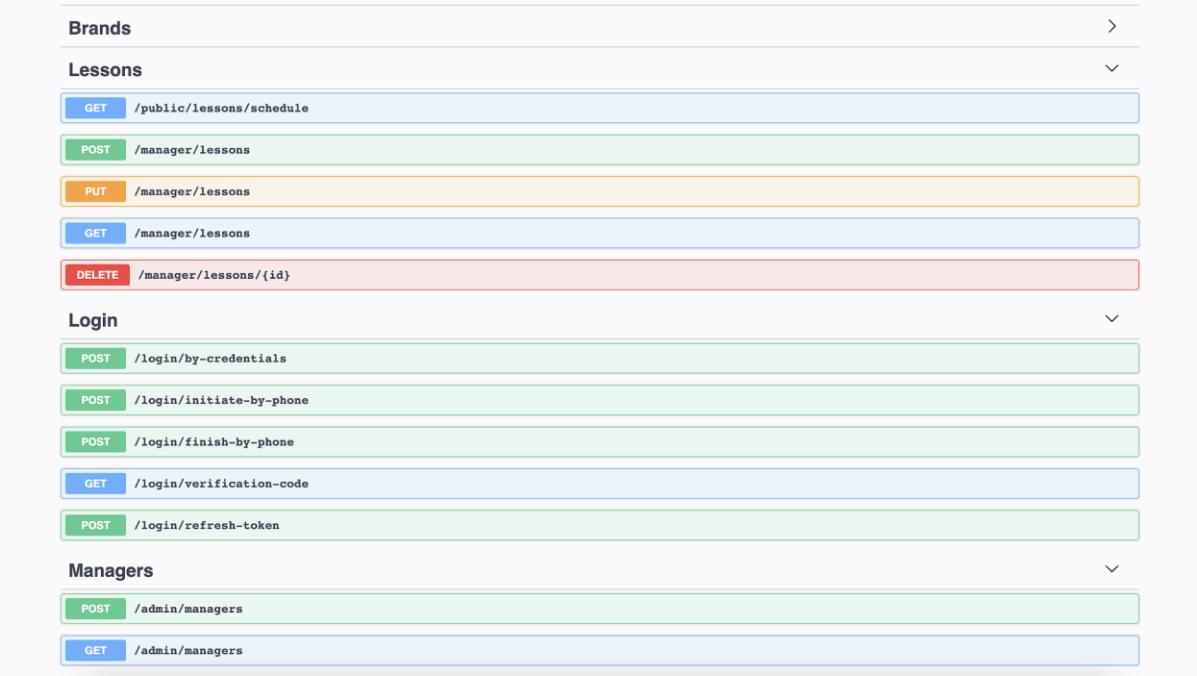

Рисунок 4.3 – Автоматично згенерована документація для CRM DOM API

В документації можна побачити всі доступні точки взаємодії. В наведеному прикладі зображено точку взаємодії для оновлення номеру телефона відвідувача танцювального центру. З документації можна побачити Uniform Resource Locator (URL), на який потрібно робити запит, щоб викликати саме цю точку взаємодії. Також наведено приклад вхідних параметрів, які очікуються на цій точці взаємодії. Та є модель відповіді, яку може повернути ця точка взаємодії. Такий детальний опис кожної з доступних точок

взаємодії полегшує використання серверного застосунку та прискорює розробку системи.

В двох застосунках, які входять до складу системи, клієнтські частини різні. Для DOM CRM користувач взаємодіє з системою через веб-сайт. А у випадку з застосунком DOM доступ до системи забезпечено через мобільний застосунок.

4.2 Імплементація підходу CQRS

В обох застосунках, які є в системі, серверні частини реалізовані з використанням підходу CQRS. Його опис можна знайти в підрозділі 3.5, а алгоритм обробки запитів з використанням цього підходу зображено в додатку Б. Суть CQRS полягає в розділені обробки команд та запитів в системі. В контексті цього підходу команди - запити, які змінюють стан системи. До найбільш типових команд відносять додавання, редагування та видалення сутностей в системі. Для отримання інформації з системи використовують запити. Тобто відмінність між запитами та командами полягає в тому, що у запитів є значення, яке вони повертають.

В системі команди та запити реалізовують інтерфейси ICommand та IQuery відповідно. Вони в собі містять інформацію, необхідну для обробки. Тому вони не мають ніяких методів в середині. Ці інтерфейси використовуються для реалізації поліморфізму. Приклад команди наведено на рисунку 4.4.

```
public class CreateStudentCommand : ICommand
{
    public Guid BrandId { get; set; }
    public string PhoneNumber { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

Рисунок 4.4 – Команда створення відвідувача танцювального центру

Ця команда використовується для створення відвідувача танцювального центру. В коді сутність відвідувача танцювального центру називається Student. Тому за клас команди називається CreateStudentCommand. Наведена команда містить всю інформацію, необхідну для створення відвідувача танцювального центру, а саме - ідентифікатор бренду, номер телефону, ім’я та прізвище клієнта. Потім дані з команди будуть використанні в її обробнику. З назви команди можна зрозуміти, що буде зроблено в системі після її виконання.

Оскільки запити використовуються для отримання інформації, то вони відрізняються від команд значенням, яке повертають. В системі для реалізації запитів створено два інтерфейси. Перший називається IQuery. Він призначений для того, щоб передавати інформацію яка вкаже які саме дані потрібно повернути у відповідь. Приклад моделі запиту наведено на рисунку 4.5.

```
public class AllBrandRanksQuery : IQuery<AllBrandRanksResponse>
{
    public Guid BrandId { get; set; }
}
```

Рисунок 4.5 – Запит типів абонементів танцювального центру

Прикладом моделі запиту є AllBrandRanksQuery. Він містить інформацію про бренд, типи абонементів якого потрібно повернути у відповідь. Параметром, який передається у інтерфейс IQuery є AllBrandRanksResponse. Це є тип, що містить інформацію, яка повертається у відповідь.

На рисунку 4.6 зображено приклад відповіді на запит підходу CQRS. Відповідь містить список об’єктів, в яких є інформація про ідентифікатор типу абонемента, назву, позицію та ціну.

```
public class AllBrandRanksResponse : IQueryResult
{
    public IList<Rank> Ranks { get; set; }

    public class Rank
    {
        public Guid Id { get; set; }
        public string Title { get; set; }
        public uint Order { get; set; }
        public decimal Price { get; set; }
    }
}
```

Рисунок 4.6 – Відповідь на запит абонементів бренду

Щоб виконати команду або запит в системі присутні їхні обробники. Вони є обов'язковими для кожної з реалізованих команд та запитів. Якщо при виконанні команди не буде її обробника, система поверне помилку виконання запиту. Кожен обробник реалізує інтерфейс ICommandHandler для команд, та IQueryHandler для запитів. Цей інтерфейс містить лише один метод Execute, який виконує команду або запит. Саме в тут, в обробнику, відбувається взаємодія з базою даних або сторонніми сервісами.

Інтерфейс ICommandHandler в якості параметрів приймає команду, в якій є інформація, необхідна для змін. Приклад обробника команди на створення відвідувача танцювального центру зображено на рисунку 4.7. В якості вхідного параметра в методі Execute приймається команда CreateStudentCommand. В самому методі відбувається взаємодія з базою даних, перевіряється чи існує користувач з вказаним брендом та номером. У випадку якщо існує, то повертається відповідна помилка. Якщо такого користувача немає в системі, то створюється новий з даними, які були передані в команді.

```
public class CreateStudentCommandHandler : CommandHandlerBase<CreateStudentCommand>
{
    public CreateStudentCommandHandler(IDOMDbContext dbContext) : base(dbContext)
    {
    }

    public override async Task Execute(CreateStudentCommand command)
    {
        var studentAlreadyExists = await DbContext.Students.AnyAsync(s =>
            s.BrandId == command.BrandId && s.PhoneNumber == command.PhoneNumber);

        if (studentAlreadyExists)
        {
            throw new InvalidOperationException("Student already exists");
        }

        var student = new Student
        {
            BrandId = command.BrandId,
            FirstName = command.FirstName,
            LastName = command.LastName,
            PhoneNumber = command.PhoneNumber,
            Subscription = null
        };
        DbContext.Students.Add(student);

        await DbContext.SaveChangesAsync();
    }
}
```

Рисунок 4.7 – Обробник команди на створення відвідувача танцювального центру

В запитів інтерфейс IQueryHandler містить метод Execute, який в якості вхідних параметрів приймає реалізації інтерфейсу IQuery, а повертає реалізації інтерфейсу IQueryResult. Суть цього метода полягає в отриманні потрібної інформації та поверненні її у відповідь в очікуваному форматі. Це все відбувається в обробнику запитів, який реалізовує інтерфейс IQueryHandler.

На рисунку 4.8 наведено приклад обробника запиту типів абонементів бренду. В методі Execute цього обробника з бази даних отримуємо всі бренди відфільтровані за ідентифікатором бренду, який прийшов у вхідному параметрі моделі запиту. Після фільтрації список типів абонементів сортується по позиції. І останній крок - дані

отримані з бази даних приводяться до потрібного формату. Відфільтровані, посортовані та в правильному форматі дані повертаються з метода.

```
public class AllBrandRanksQueryHandler : QueryHandlerBase<AllBrandRanksQuery, AllBrandRanksResponse>
{
    public AllBrandRanksQueryHandler(IDOMDbContext dbContext) : base(dbContext)
    {
    }

    public override async Task<AllBrandRanksResponse> Execute(AllBrandRanksQuery query)
    {
        var ranks = await DbContext.Ranks
            .Where(rank => rank.BrandId == query.BrandId)
            .OrderBy(rank => rank.Order)
            .Select(rank => new AllBrandRanksResponse.Rank
            {
                Id = rank.Id,
                Order = rank.Order,
                Title = rank.Title,
                Price = rank.Price
            })
            .ToListAsync();

        return new AllBrandRanksResponse{Ranks = ranks};
    }
}
```

Рисунок 4.8 – Обробник запиту типів абонементів бренду

Правильна реалізація підходу CQRS забезпечує дотримання принципу Single responsibility з SOLID. Оскільки для кожної команжи або запиту існує свій обробник, то він виконує лише одну дію. Тому він матиме одну причину для змін. Це робить програмний код більш зрозумілим, а структуру проекту - простішою.

В системі існує багато команд та запитів, і відаовідно багато обробників. Щоб не викликати обробники на пряму, в системі використано поліморфізм. Він забезпечує гнучкість та дає можливість розширити систему без змін у вже існуючому функціоналі. Його реалізовано в CommandBus та QueryBus. Їхня реалізація дуже схожа, тому можна розглянути лише на одному прикладі, який наведено на рисунку 4.9.

```
public class CommandBus : ICommandBus
{
    private readonly IServiceProvider _serviceProvider;

    public CommandBus(IServiceProvider serviceProvider)
    {
        _serviceProvider = serviceProvider;
    }

    public async Task Execute<TCommand>(TCommand command) where TCommand : ICommand
    {
        var validator = _serviceProvider.GetService<IValidator<TCommand>>();

        if (validator != null)
        {
            var validationResult = await validator.ValidateAsync(command);

            if (!validationResult.IsValid)
            {
                throw new ValidationException(validationResult.Errors);
            }
        }

        var commandHandler = _serviceProvider.GetService<ICommandHandler<TCommand>>();

        if (commandHandler == null)
        {
            throw new ArgumentException("Handler for specified type of command was not found", nameof(command));
        }

        await commandHandler.Execute(command);
    }
}
```

Рисунок 4.9 – Реалізація CommandBus

Він використовується для того, щоб не викликати обробники команд на пряму. Його призначення - визначити обробник команди та передати її на виконання. Він містить лише одни метод - Execute. Вхідним параметром є модель команди, яку потрібно виконати. З контейнера залежностей отримуємо валідатор для команди. Якщо він існує, валідуємо дані в команді. Якщо вони не валідні, повертаємо відповідну помилку. Після валідації команди з контейнера залежностей отримуємо обробник команди. Якщо його не існує, також повертаємо відповідну помилку. Саме тому обробник є обов’язковим для кожної команди. Така ж ситуація із запитами. Після того як обробник знайдено, викликаємо в нього метод Execute, описаний раніше, та передаємо в якості вхідного

параметру модель команди. Відмінність в QueryBus полягає в тому, що з обробника очікуємо отримати результат, який повертаємо.

Така реалізація використовує поліморфізм. Адже коли ми викликаємо метод Execute в обробника, ми не знаємо в якого саме. Це робить систему дуже гнучкою. З'являється можливість використовувати загальний механізм виклику всіх команд та запитів, додавати нові команди та запити і, не змінюючи існуючого коду, викликати їх через CommandBus.

Все це є можливим через інтерфейси. В CommandBus отримується не конкретний обробник, а інтерфейс ICommandHandler, який є загальним для всіх обробників. Користуючись цим інтерфейсом, в ході виконання програми виконуються дії в конкретному обробнику.

Приклад використання CommandBus та QueryBus є дуже простий. Він наведений на рисунку 4.10. Полягає він у виклиці метода Execute та передачі в якості вхідного параметра моделі команди або запиту.

```
public class CqrsController : ControllerBase
{
    public ICommandBus CommandBus { get; }
    public IQueryBus QueryBus { get; }

    public CqrsController(ICommandBus commandBus, IQueryBus queryBus)
    {
        CommandBus = commandBus;
        QueryBus = queryBus;
    }

    protected Task Execute<TCommand>(TCommand command) where TCommand : ICommand
    {
        return CommandBus.Execute(command);
    }

    protected Task<TResult> Execute<TQuery, TResult>(TQuery query)
        where TQuery : IQuery<TResult>
        where TResult : IQueryResult
    {
        return QueryBus.Execute<TQuery, TResult>(query);
    }
}
```

Рисунок 4.10 – Приклад використання CommandBus та QueryBus

4.3 Розробка бази даних та взаємодія з нею

В створеній системі використовується СУБД MSSQL. Її було описано в підрозділі 3.4. Це реляційна база даних, тому дані зберігаються в таблицях і мають зв’язки між собою. Взаємодія з нею відбувається через ORM Entity Framework. Переваги, недоліки та принцип використання цієї технології були описані в підрозділі 3.4.

В системі існує контекст бази даних, через який і відбувається безпосередня взаємодія з джерелом даних. Сутності, які доступні через нього представлені властивостями контексту. Вони мають тип DbSet і зображені на рисунку 4.13.

```
public class DOMDbContext : DbContext, IDOMDbContext
{
    public static readonly string ConnectionStringName = nameof(DOMDbContext);

    public DbSet<UserBase> Users { get; set; }
    public DbSet<Teacher> Teachers { get; set; }
    public DbSet<Student> Students { get; set; }
    public DbSet<Manager> Managers { get; set; }
    public DbSet<Employee> Employees { get; set; }
    public DbSet<Admin> Admins { get; set; }
    public DbSet<Brand> Brands { get; set; }
    public DbSet<Lesson> Lessons { get; set; }
    public DbSet<Rank> Ranks { get; set; }
    public DbSet<Room> Rooms { get; set; }
```

Рисунок 4.13 – Контекст бази даних

В контексті оголошено 10 сутностей, проте таблиць в базі даних є 6. Справа в тому, що кілька сутностей знаходяться в одній таблиці. Обґрунтуванням такого способу імплементації є те, що в цих сутностей більшість властивостей однакові. В коді це реалізовано за допомогою визначення потрібної конфігурації в методі OnModelCreating. Його зображено на рисунку 4.14.

```
modelBuilder.Entity<UserBase>()
    .HasDiscriminator(user => user.Type)
    .HasValue<Admin>(UserType.Admin)
    .HasValue<Manager>(UserType.Manager)
    .HasValue<Employee>(UserType.Employee)
    .HasValue<Student>(UserType.Student)
    .HasValue<Teacher>(UserType.Teacher);
```

Рисунок - 4.14 – TPH конфігурація

Оскільки сутності Admin, Manager, Employee, Student та Teacher наслідуються від класу UserBase, то EntityFramework потрібно вказати яким чином цю ієрархію переносити в СУБД. В системі використано підхід Table Per Hierarchy (TPH). Його зміст полягає в тому, що на всю ієрархію класів в C# коді створюється одна таблиця, яка містить колонки для всіх типів. Щоб була можливість в коді відрізняти до якого типу відносяться дані з таких таблиць, в них створюється додаткова колонка, яка виступає дискримінатором. В системі DOM вказано, що дискримінатором в таблиці користувачів є колонка Type. В конфігурації контексту бази даних вказано які дані в неї записувати для кожного з доступних типів користувачів.

Окрім конфігурації TPH при конфігурації бази даних, також слід звернути увагу на те, що система є спільною для всіх танцювальних центрів. Тобто дані різних брендів будуть зберігатись в одній базі даних. Вид такої архітектури системи називається multi-tenant. Для її реалізації було додано колонку BrandId у таблиці, дані з яких мають прив’язку до бренду.

Візуально структура бази даних зображена в додатку В. Таблиця Brands являє собою сутність танцювального центру. В ній присутні наступні колонки:

- ідентифікатор;
- назва танцювального центру;
- адреса електронної пошти власника танцювального центру;

- дата та час створення бренду в системі;
- прапорець чи був видалений цей бренд;

Ідентифікатор виступає первинним ключем в цій таблиці і використовується, щоб мати доступ до запису про бренд по певному унікальному значенні. Назва танцювального центру являє собою строку та відображається різним типам користувачам відповідного танцювального центру. Прапорцем у випадку бази даних є булеве значення. Тут воно свідчить про те, чи був видалений танцювальний бренд з системи. Такий механізм видалення називається soft delete. Його використано в цій системі для зменшення кількості помилок та відповідно прискорені розробки системи. Віз забезпечує те, що не буде посилань на інформацію, яка була видалена з бази даних.

Дані про типи абонементів, доступних в танцювальному зберігаються в таблиці Ranks. В ній є такі колонки:

- ідентифікатор;
- ідентифікатор танцювального центру;
- назва;
- порядок;
- ціна;

Ідентифікатор є первинним ключем в таблиці і використовується як для посилання на записи з даної таблиці. Ідентифікатор танцювального центру вказує на запис з вищезгаданої таблиці брендів до якого цей тип абонемента належить. Кожен тип абонементу має свою назву. Порядок типу абонементу представлений звичайним числом. Приклад його використання наступний. Відвідувач танцювального центру, придбавши більш дорогий абонемент, має змогу відвідувати заняття викладачів, які доступні при більш дешевому абонементі. В таких випадках в коді порівнюється порядок типу абонемента наявного у відвідувача та тип абонементу необхідний, що відповідає вчителю, який веде заняття. Якщо порядок першого абонементу більший, то

відвідувач має змогу відвідати урок цього викладача. Також в цій таблиці присутня колонка з ціною відповідного типу абонемента.

Кожен танцювальний центр має зали, в яких проводяться заняття. Інформація про них зберігається в таблиці Rooms. Про кожну залу міститься наступна інформація:

- ідентифікатор;
- ідентифікатор бренду;
- назва;

Ідентифікатор зали слугує первинним ключем і використовується для посилання на потрібний запис з цієї таблиці. Ідентифікатор бренду використовується для зв'язку з танцювальним центром, до якого ця зала відноситься. Назва бренду використовується для зручності користувачів.

Інформація про заняття, які проходять в танцювальному центрі, зберігається в таблиці Lessons. Кожен урок має такі властивості:

- ідентифікатор;
- ідентифікатор бренду;
- ідентифікатор залу;
- ідентифікатор вчителя;
- день тижня;
- час початку заняття;
- час закінчення заняття;
- опис;

Ідентифікатор є первинним ключем уроку, на який можуть мати посилання інші таблиці. Ідентифікатор бренду вказує на танцювальний центр, в якому цей урок проходитиме. Ідентифікатор залу посилається на запис з таблиці Rooms. Це вказує на зал танцювального центру, де цей урок буде проходити. Ідентифікатор вчителя - посилання на таблицю Teachers. Він вказує на вчителя, який буде вести це заняття. День тижня в базі даних записується числовим значенням і відповідає порядковому номеру

дня тижня, коли цей урок відбуватиметься. В коді кожному дню тижня відповідає значення з переліку. В створеній системі використано власне перерахування, а не те, яке надається в .NET. Причиною цьому є те, що в створеному переліку містяться прапорці, за допомогою яких можна одночасно вказати на кілька значень з нього. Так можна вказати на всі будні дні. Окрім цього кожен урок містить інформацію про час початку та завершення заняття. Також врахована потреба бренду вказувати опис до кожного заняття.

Як згадувалось раніше сутності Admin, Manager, Employee, Student та Teacher зберігаються в одній таблиці. Причиною цьому є спільність багатьох властивостей. Таке рішення має недолік у вигляді надлишкових колонок для певних типів користувачів. Це суперечить другій нормальній формі бази даних, оскільки не всі атрибути використовуються для кожного із записів у таблиці. Проте цей недолік нівелюється перевагою у вигляді швидкості розробки системи. Система не є великою, тому оптимальним рішенням була така імплементація. Адже це зекономило час, необхідний для створення ще чотирьох таблиць. Вищезгадані сутності зібрані в таблиці Users, яка має наступну інформацію:

- ідентифікатор;
- ім’я;
- прізвище;
- адреса електронної пошти;
- пароль в зашифрованому вигляді;
- ідентифікатор бренда;
- тип;
- номер телефону;
- refresh token;
- кількість доступних занять;
- дата припинення дії абонемента;

- ідентифікатор типу абонемента;
- загальна кількість занять, доступних по абонементу;
- код верифікації;
- токен верифікації;
- прапорець чи був абонемент оплачений;

Первинним ключем в цій таблиці виступає ідентифікатор. Він використовується для посилання на записи з цієї таблиці. Ім’я та прізвище користувачів різних типів зберігаються у одноіменних колонках. Адреса електронної пошти користувача та його пароль в зашифрованому вигляді використовуються для авторизації деяких типів користувачів. Ідентифікатор бренда використовується для зв'язку між користувачем та танцювальним брендом, до якого він відноситься. Для того, щоб відрізняти типи користувачів створено колонку, в якій вказано до якого типу відноситься цей користувач. В ній зберігаються чисельні значення, які відповідають переліку типів користувачів, який є в коді. Телефон користувача зберігається в однойменній колонці і використовується для авторизації певних типів користувачів. У відповідну колонку записується refresh token, коли він формується при будь-якій з видів авторизацій доступних в системі. Також в цій таблиці наявна інформація про абонемент користувача, якщо такий існує, а саме доступна кількість занять, дата припинення дії, тип, загальна кількість занять, яка доступна за цим абонементом та чи був він оплачений. Для авторизації в таблиці також є колонки для коду та токену верифікації.

В системі один вчитель може відповідати декільком типам абонементів. Це можливо, коли викладач навчає різним стилям танцю. Тому в системі реалізовано зв’язок багато до багатьох між вчителями та типами абонементів. Зроблено це з використанням проміжної таблиці TeacherRank. В ній містяться лише дві колонки, кожна з яких є частиною складеного первинного ключа. Перша колонка – ідентифікатор вчителя, друга колонка – ідентифікатор типу абонемента.

Крім цих таблиць в системі присутня ще одна. Вона називається __EFMigrationsHistory та використовується Entity Framework'ом для можливості підключення бази даних до заданої міграції. В ній присутні лише дві колонки:

- ідентифікатор міграції;
- версія продукту;

Entity Framework використовує цю таблицю при маніпуляціях з міграціями бази даних. Коли нова міграція створюється, в цю таблицю додається новий запис. Після є можливість повернутися на будь-яку попередню міграцію і створити нову, перейшовши одразу на неї. Тобто зміни міграцій, які були обійдені не будуть застосовані до бази даних.

4.4 Реалізація процесів авторизації

В системі в обох застосунках процес авторизації виконано з дотриманням стандарту OAuth 2.0. А для формування access token'а використано стандарт JWT. Принцип роботи цих стандартів було описано в підрозділі 3.7, а в цьому підрозділі наведено спосіб їх реалізації. В кожному із застосунків вони були імплементовані по-різному для зручності використання. Але на виході, після процесу авторизації результат однаковий - користувач авторизований в системі. А в контексті коду - користувача отримав свої access token та refresh token, використовуючи які може виконувати авторизовані запити в системі.

4.4.1 Реалізація процесу авторизації у веб-застосунку

У веб застосунку DOM CRM користувачі взаємодіють з системою через браузер. Тому тут використано звичайний спосіб авторизації з використанням адреси електронної пошти та пароля. Цей алгоритм є простішим, ніж той, який

використовується в іншому застосунку, оскільки в ньому присутні менше кроків для авторизації користувача.

Процес авторизації у веб-застосунку починається з сторінки логіна на клієнтській частині, яка зображена на рисунку 4.15. Тут користувачу потрібно ввести свої пароль та адресу електронної пошти, які він вказував при реєстрації.

Після підтвердження форми виконується запит на серверну частину. Там він потрапляє в сервіс, який повертає пару access token та refresh token. Принцип роботи цього сервіса простий. Він полягає в знаходженні користувача в системі за введеними адресою електронної пошти та паролем.

Слід зазначити, що в системі паролі зберігаються в зашифрованому вигляді, щоб якщо зловмисник отримає доступ до них, він не міг ними скористатись. Для цього використано стандарт Password-Based Key Derivation Function 2, оскільки це було однією з вимог до системи. Якщо користувача з введеними даними не знайдено, повертається відповідна помилка, і він залишається не авторизованим в системі. В іншому випадку процес авторизації продовжується формуванням access token’а та генерацією refresh token’а.

Для того щоб створити access token, за адресою електронної пошти та зашифрованим паролем отримується ідентифікатор користувача в системі. Потім він використовується для формування корисної частини JWT токена. Також до корисної частини входить і дата закінчення терміну придатності токена.

В наступних запитах до системи, які вимагають авторизації, ці дані дешифруються з access token’а і використовуються. Коли корисна частина сформована, вона підписується секретним ключем, який зберігається лише на сервері і шифрується в base64 строку. На виході утворюється access token. Для формування refresh token’а використано сервіс генерації рандомних рядків. Для його використання у вхідні параметри йому потрібно передати кількість символів, яка повинна бути в отриманому

рядку. В системі ця інформація зберігається в файлі конфігурації,тому легко може бути змінена.

Після створення токенів вони повертаються на клієнтську частину застосунку. Там вони зберігаються в local storage для подальшого їх використання при авторизованих запитах на серверну частину системи.

4.4.2 Реалізація процесу авторизації в мобільних застосунках

В мобільному застосунку DOM процес авторизації відрізняється від того, який реалізовано в веб-застосунку. Причиною цьому є тип пристрою з якого користувач отримує доступ до системи. Оскільки мобільний застосунок доступний лише зі смартфона, тож логічно припустити, що користувачеві зручніше буде використати номер телефону для авторизації в системі. Також авторизація по мобільному номеру телефона є більш безпечною, оскільки при кожній спробі авторизуватись перевірка особистості користувача відбувається з використанням смартфону. Візуально послідовність авторизації таким способом зображено на відповідній діаграмі послідовності в додатку Г.

Такий процес авторизації починається з екрану авторизації в мобільному застосунку. На ньому зображено лише одне поле для вводу даних, де користувачеві потрібно ввести номер телефону, який було вказано при реєстрації. Важливо, щоб мобільний пристрій з цим номером був доступний користувачу, який авторизується.

Екран авторизації в мобільному застосунку зображено на рисунку 4.16. На ньому представлено вигляд одразу на двох найпопулярніших мобільних операційних система – Android та iOS. Це показує перевагу технології Xamarin, адже код для відображення та логіка застосунку написана один раз і застосовується для двої ОС.

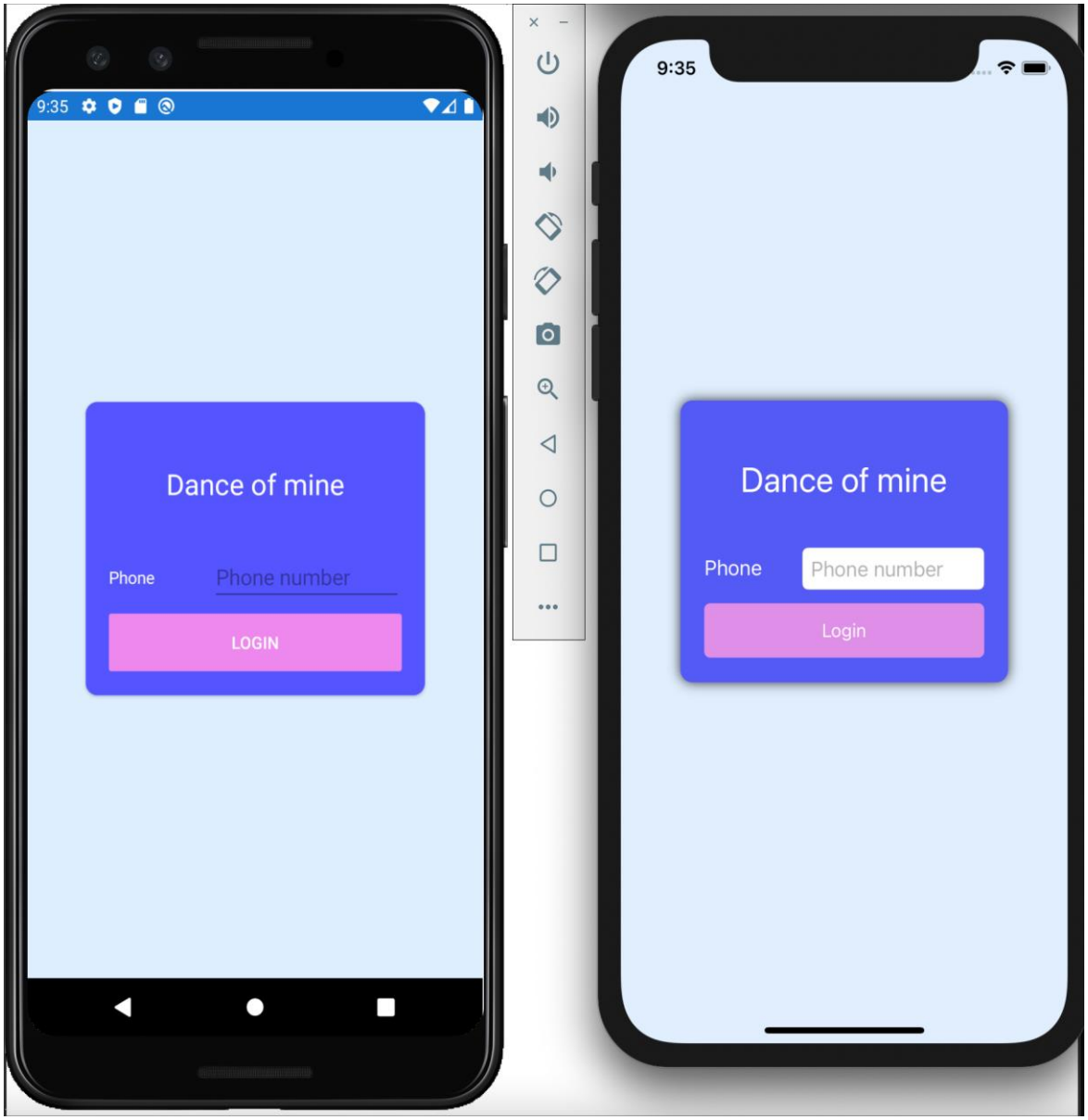

Рисунок 4.16 – Екран авторизації в мобільному застосунку DOM

Також в Xamarin є можливість визначати вигляд сторінок застосунків та бізнес-логіку для кожної з операційних систем окремо. Для цього в проекті застосунку є проекти під кожну ОС. Проте в створеній системі не має необхідності таке робити, тому вся бізнес-логіка прописана вспільному проекті.

Після того як користувач ввів номер телефону та натиснув кнопку “Login”, користувач потрапляє на екран підтвердження номеру телефону. Там він бачить також

єдине поле для вводу даних та кнопку “Submit”. На цій сторінці йому слід ввести шестизначний код, який прийде у СМС на вказаний раніше номер.

Коли користувач введе правильний код верифікації та натисне кнопку “Submit”, він потрапитьголовну сторінку мобільного застосунку. На цей момент він вже є авторизованим в системі, тобто може отримувати інформацію з точок доступу, в яких необхідна авторизація.

Для реалізації авторизації за допомогою коду з СМС повідомлення, в коді передбачається інтеграція з стороннім сервісом надсилання СМС повідомлень. Після підтвердження номеру телефону та його успішної валідації на першій сторінці, виконується запит на серверну частину застосунку. Там виконується формування коду верифікації, який відсилається на сервіс дасилання СМС повідомлень. Після цього користувачу приходить повідомлення з цим кодом верифікації на номер вказаний на першій сторінці.

Далі, коли користувач вводить код верифікації, в мобільному застосунку. В коді виконується запит на серверну частину, в якому передається код верифікації та токен. Токен верифікації являє собою JWT токен, в якому міститься інформація про ідентифікатор користувача та дату й час завершення дії токена. Він повертається у відповідь на запит, в якому передається номер телефону. Токен зберігається зберігається в мобільному застосунку для подальшого використання і не видний кінцевому користувачу мобільного застосунку.

Потім з токена отримується корисна інформація, яка була схована в нього при формуванні. Ця інформація валідується і у випадку її валідності відбувається пошук користувача в системі по ідентифікатору, отриманому з токена. Якщо такого користувача знайдено генерується refresh token та access token. Refresh token зберігається в базу даних. Після два токени повертаються у відповідь на клієнтський застосунок. Там вони зберігаються в кеш і передаються в запитах, що вимагають автентифікації.

4.5 Висновок

В розділі наведено опис розробленої частин системи та підходів, використаних в ній.

В розділі 4.1 описано яким чином створювалась клієнт-серверна архітектура. Також вказано яким чином працює система за такої архітектури.

Розділ 4.2 містить опис підходу CQRS та детально описану його реалізацію в системі.

В розділі 4.3 міститься інформація про те, яким чином використано технології обрані для збереження даних. Наведено опис структури бази даних.

В розділі 4.4 наведено інформацію про клієнтські застосунки, які входять до складу системи та те, як вони були розроблені.

5 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

5.1 Принцип роботи системи

Для формування принципу, за яким буде працювати система не можна було повністю відштовхуватись від огляду існуючих рішень, проведеного в розділі 1.2. Причиною цього є факт того, що оглянуті CRM-системи мають загальне призначення для бізнесу, а не заточені виключно під тематику танцювально центру. Тому також бралась до уваги специфіка предметного середовища, описана в розділі 1.1.

Було прийнято рішення додати можливість керувати розкладом занять в танцювальному центрі. Такої можливості немає в рішеннях-аналогах, проте вона є корисною для забезпечення доступу до інформації в електронному вигляді. Також прийнято рішення зробити кілька видів користувачів для працівників танцювальних центрів. Оскільки в існуючих рішеннях не враховано функціональність з розкладом занять, тому в них немає видів користувачів, які відповідають тим, які є в танцювальних центрах.Це дозволить ефективно поділити ролі для працівників бренду, таким чином оптимізувати роботу.

Як результат було створено систему танцювального центру Dance Of Mine (DOM), яка складається з кількох застосунків, що призначені різним типам користувачів.

Перший застосунок називається DOM CRM. Він створений для використання менеджерами та адміністраторами танцювального центру. Також він використовується адміністраторами цілої системи. Другий застосунок називається DOM і призначений для клієнтів та викладачів танцювальних центрів.

В системі можна виокремити користувачів, які пов’язані з конкретним танцювальним центром, та працівників самої системи DOM. Сценарії використання користувачів, які відносяться до конкретного бренду, описано в розділі 5.2. В цьому розділі розглянемо принцип використання системи з точки зору адміністратора системи DOM.

Задачею адміністратора системи DOM є налаштування системи для використання працівниками танцювального центру. Він створює бренд - так в системі називається обліковий запис танцювального центру. Для цього він використовує форму вказану на рисунку 5.1.

Після того як створено бренд танцювального центру адміністратор створює його менеджерів. Для цього адміністратор вказує персональні дані менеджера, серед яких адреса його поштової скриньки. Після створення менеджера, на вказану адресу прийде повідомлення, в якому буде міститися посилання. Перейшовши за цим посиланням буде запропоновано встановити пароль до свого облікового запису. Після цього, його обліковий запис буде сформований і він матиме змогу авторизуватись в системі.

5.2 Сценарії використання застосунків у системі

В системі присутньо кілька застосунків. Перший має назву DOM CRM і створений для внутрішнього використання адміністраторами системи та в танцювальних центрах. Він являє собою веб-застосунок, взаємодія з яким відбувається через браузер. Інший застосунок називається DOM та призначений для використання вчителями танцювального центру та його відвідувачами. Він являє собою мобільний застосунок для більш зручного використання кінцевими користувачами.

В описі сценаріїв використання системи бренд є танцювальним центром. Адміністраторами системи називають працівників, які додають новий бренд в систему та створюють його менеджерів. Менеджерами називають працівників танцювального центру, які будуть мати доступ до керування ним. Адміністратори бренду є також його працівниками, проте з меньшими правами в системі. Вони є персоналом, який обслуговує відвідувачів танцювального центру на рецепшині. Також в системі є інші користувачі - вчителі та відвідувачів танцювального центру. Можливості всіх вищезазначених видів користувачів описані в додатку Д.

5.2.1 Сценарії використання веб застосунку

Для керування процесами танцювального центру та усіма брендами, які є в системі, створено застосунок DOM CRM. Він призначений для використання адміністраторами системи та конкретних танцювальних центрів. Для початку роботи з DOM CRM користувача спочатку потрібно авторизуватись. Авторизація в застосунку відбувається через введення логіна та пароля, сама форма зображена на рисунку 5.2. Вона однакова для адміністраторів системи та працівників танцювального центру.

Рисунок 5.2 – Форма авторизації в DOM CRM

Після авторизації процес взаємодії для адміністраторів системи та працівників танцювального центру різний, тому їх слід розглядати окремо.

Адміністратори системи після авторизації потрапляються на сторнку зі списком всіх брендів, які є в системі. Тут відображається назва бренду та адреса електронної пошти власника, рисунок 5.3.

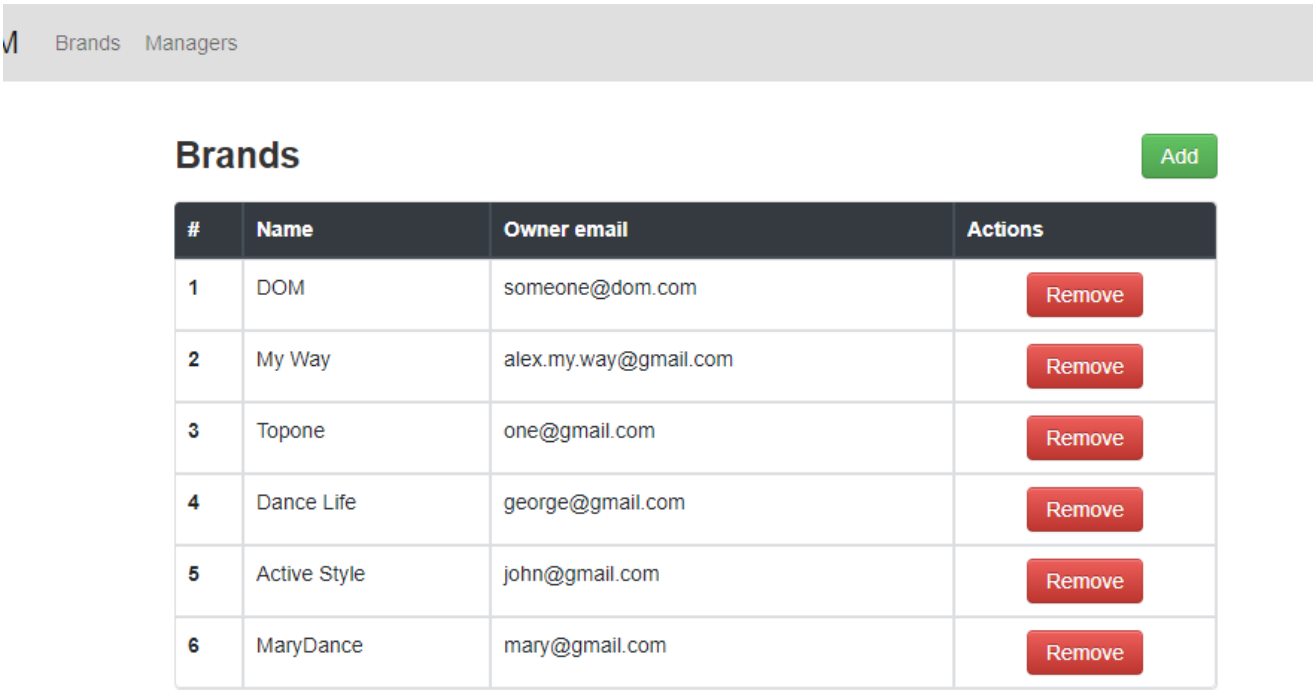

Рисунок 5.3 – Список брендів в системі

На цій сторінці адміністратор має змогу видалити вже існуючий бренд. Для цього йому слід натиснути кнопку “Remove” в рядку з відповідним брендом. Після чого в нього з'явилася модальне вікно, в якому йому потрібно буде підтвердити свою дію.

Також на сторінці зі списком всіх брендів адміністратор має змогу створити новий бренд. Для цього йому варто натиснути кнопку “Add”. Вона знаходиться над таблицею зі всіма брендами в правому верхньому куті сторінки. Після того як адміністратор натисне цю кнопку, на екрані з'явиться форма для створення бренду, зображена на рисунку 5.4. Вона відображається як модальне вікно, тобто переходу на нову сторінку не відбувається. Заповнивши її, адміністратор повернеться на сторінку зі списком брендів, де буде відображено новостворений запис.

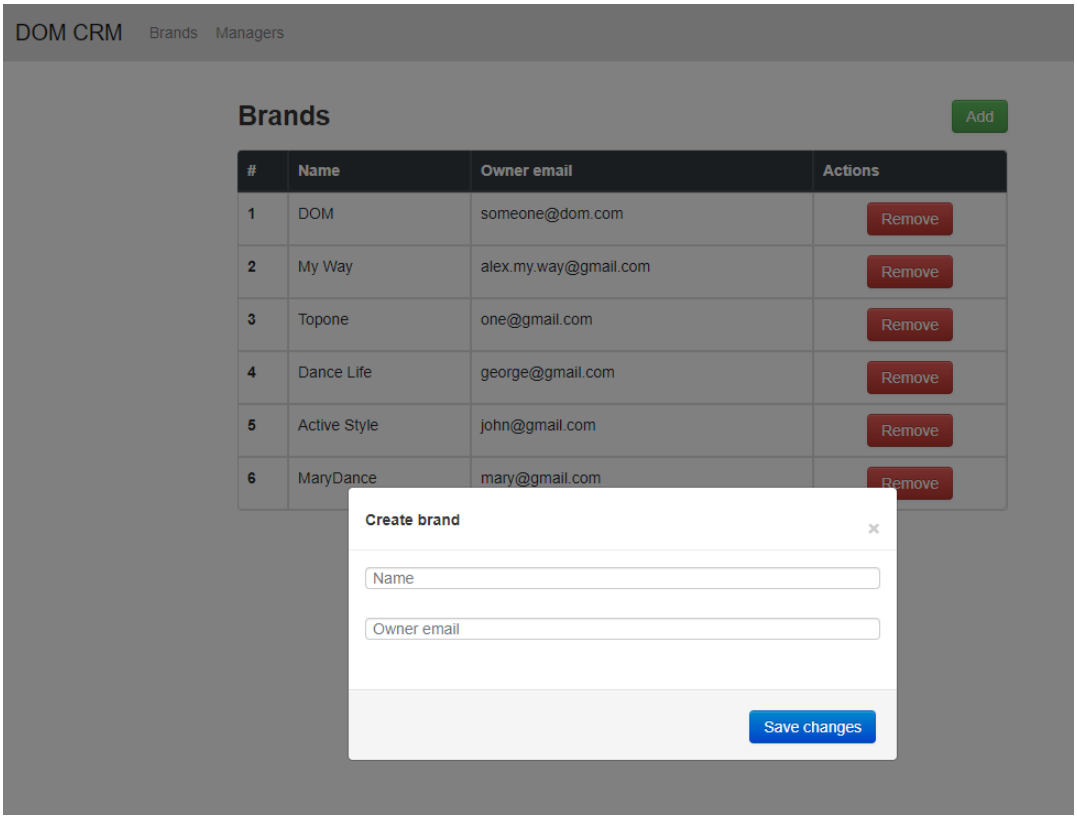

Рисунок 5.4 – Форма створення нового бренду

Натиснувши на бренд, адміністратор перейде на сторінку з аналітикою та детальною інформацією про бренд. На цій сторінці всі дані будуть згруповані по вкладках.

Вкладка “Students” буде містити загальну кількість учнів. Також буде доступне абсолютне та відсоткове значення розподілу по групах абонементів.

Також адміністратору буде доступна вкладка з інформацією про абонементи - “Subscriptions”. Тут він зможе переглянути загальну виручку танцювального центру та виручку з кожного тип абонемента

Інформація про вчителів також буде доступна на цій сторінці. Вона буде знаходитись в окремій вкладці “Teachers”. Тут буде міститись перелік усіх викладачів та відсоткову частку студентів, які відвідали їхні заняття за останній тиждень.

Окрім інформації про бренд на цій сторінці також буде список всіх менеджерів танцювального центру. Вона буде знаходитись в окремій вкладці “Managers”. В списку

буде відображатись ім’я та прізвище працівника та його електронна адреса. Навпроти кожного запису буде кнопка “Remove” для видалення облікового запису відповідного менеджера з бренду танцювального центра.

Повертаючись до працівників танцювального центру, то після авторизації в системі вони потрапляють на різні сторінки. У випадку з менеджером танцювального центру, він потрапляє на сторінку з детальною інформацією про бренд. Ця сторінка містить інформацію поділену по вкладках.

Вкладка “Students” містить інформацію про студентів. Тут відображено поділ студентів по групах наявних абонементів. Які і в адміністратора системи, в менеджера відображається абсолютне та відсоткове значення. Також тут доступна загальна кількість студентів. На відміну від адміністратора системи, менеджер може переглянути інформацію про студента, натиснувши у відповідному рядку.

На вкладці “Subscriptions” знаходитиметься інформація про абонементи танцювального центру. Тут вказано суму коштів, на яку було куплено абонементи кожної з груп. Також тут міститься інформація про загальну виручку з продажі всіх видів абонементів.

Ще однією доступною вкладкою є “Teachers”. Тут менеджеру танцювального центру відображається список усіх вчителів цього бренду. Кожен запис в списку містить ім’я та прізвище, тип абонемента та відсоткову частку студентів, які відвідали заняття цього вчителя за останній тиждень. Натиснувши на одного з них, менеджер перейде на сторінку з детальною інформацією про викладача. Також у менеджера є можливість видалити вчителя, натиснувши кнопку “Remove” навпроти відповідного запису зі списку.

Відмінним від адміністратора системи буде вкладка "Employees". Перейшовши на неї, менеджер танцювального центру бачить список всіх адміністраторів його бренду. В кожному рядку буде прізвище та ім'я працівника, а також кнопка "Remove" для видалення відповідного адміністратора з танцювального центру. При натисканні на

рядок менеджер перейде на сторінку детальної інформації про відповідного адміністратора.

Також на сторінці зі списком усіх адміністраторів танцювального центру буде кнопка "Add". При натисканні на неї, з'являється форма створення нового адміністратора танцювального центру. Після підтвердження форми менеджер повертається на сторінку зі списком всіх працівників бренду.

В системі є розклад занять. Можливість керувати ними надана менеджеру танцювального центру. Він це може зробити з вкладки "Lessons". Перейшовши на неї, менеджер буде бачити панель з вкладками. Кожна вкладка буде відповідати окремому залу цього бренду. Натиснувши на кнопку "Add", створиться новий зал і менеджеру буде запропоновано ввести його назву. Кількість залів в танцювальній школі необмежена.

Перейшовши на вкладку з потрібним залом, менеджер побачить таблицю з розкладом уроків. Щоб створити новий урок, слід натиснути на комірку в розкладі. Після цього з'явиться форма для створення уроку. Заповнивши її менеджер повернеться на сторінку з розкладом, де вже буде додано щойно створений урок. Якщо натиснути на урок в розкладі, відкривається форма з даними про урок. Тут можна змінити раніше введені дані. Також у формі є кнопка “Remove”, натиснувши на яку, менеджер має змогу видалити урок.

Також менеджеру доступна можливість створювати типи абонементів та визначати ціни на них. Для цього йому доступна вкладка “Ranks”. В ній відображається список існуючих типів абонементів танцювального центру. Також я кнопка “Add” для додавання нового типу абонемента. Якщо натиснути на рядку з існуючим типом абонемента, відкриється форма з наступною інформацією про нього: назва, ціна за певне кількість занять та список вчителів, які відповідають цьому типу абонемента. Тут є можливість змінити дані про абонемент та зберегти їх.

Окрім менеджера, CRM-системою також користується адміністратор танцювального центру. Слід зазначити, що всі дії, доступні адміністратору конкретного бренду, може виконувати і менеджер.

Адміністратор танцювального центру після авторизації потрапляє на сторінку з кількома вкладками. За замовчуванням йому відкривається вкладка “Main”, де він має змогу керувати абонементами користувачів. Перейшовши на неї адміністратор бачить поле для вводу номера телефону відвідувача танцювального центру, до якого прив’язаний абонемент. Після вводу даних та їх підтвердження за допомогою кнопки “Submit” адміністратору відображається інформація про користувача та абонемент.

Він має змогу змінити дату до якої абонемент залишається придатним для використання та використати певну кількість занять. Слід зазначити, що є можливість використати кілька занять за одну операцію, оскільки можливий випадок, коли клієнт танцювального центру планує відвідати заняття кількох викладачів в один вечір. Щоб не повторювати операцію списання доступних занять на абонементі, реалізовано можливість використати відразу кілька доступних занять.

В системі адміністратор танцювального центру має змогу переглянути розклад занять. Для цього йому доступна вкладка “Lessons”. На відміну від менеджерів бренду, адміністратор не має можливості змінювати інформацію тут. Ця вкладка доступна адміністратору з метою забезпечення користувачів інформацією про розклад.

5.2.2 Сценарії використання мобільного застосунку

Мобільний застосунок системи називається DOM і є призначеним для вчителів та відвідувачів танцювального центру. Авторизація в ньому для цих двох видів користувачів однакова і відбувається за допомогою номера телефону. Більш детально з цим процесом можна ознайомитись в додатку Г.

З точки зору користувача мобільного застосунку він відбувається наступним чином. Спочатку слід ввести номер телефону, який в системі присвоєний до користувача. Далі на цей номер телефону прийде СМС з кодом верифікації. Його слід ввести на наступній сторінці, яка відкриється користувачеві після вводу номера телефону. Якщо код невірний, то користувач залишиться на сторінці вводу коду, щоб спробувати ще раз. Також він може повернутися на попередню сторінку і вказати інший номер телефону. У випадку, якщо номер телефону правильний, користувач буде авторизований в системі.

Після авторизації вчителі потрапляють на екран зі своїм розкладом. Є можливість переглянути загальний розклад, вибравши відповідну позицію в контекстному меню. Розклад відображається у вигляді карток, кожна з яких представляє день тижня. На картці списком подано перелік уроків в конкретний день. В кожному рядку відображається інформація про час початку заняття, його назва та зал, в якому це заняття буде проводитися.

В мобільному застосунку для вчителя присутнє бокове меню, відкривши яке можна побачити позиції “Schedule” та “Lessons”. Натиснувши на позицію “Schedule” вчитель потрапить на сторінку, яку описано вище. При натисканні на позицію “Lessons” в боковому меню вчителю відобразиться список усіх його класів згрупованих по назві. Перейшовши на будь-яке із занять, вчитель побачить його детальну статистику за тиждень, що триває.

До статистичної інформації входить кількість людей на занятті, відсоткове співвідношення людей, які прийшли на цей клас вперше, та тих, хто ходить постійно. Також є відсоткове співвідношення кожної групи абонементів, які використовувались при оплаті заняття. Зображено відвідуваність у відсотковому співвідношенні по днях тижня. Присутнє порівняння з статистикою за минулий тиждень. Також є можливість відкрити статистичні дані за будь-який з минулих тижнів.

Інший вид користувачів мобільного застосунку, після авторизації в ньому також потрапляють на сторінку з розкладом занять танцювального центру. Розклад складається з карток, які відповідають дням тижня. В кожній з карток міститься список занять цього дня. В ньому міститься наступна інформація про кожне заняття: час початку назва, викладач та зал, в якому його буде проведено.

В учнів, як і у вчителя, присутнє бокове меню. В ньому знаходиться вищеописана позиція “Schedule” та позиція “Subscriptions”. Перейшовши на другу, відвідувач танцювального центру побачить інформацію про свій абонемент. Тут зображено дату початку та закінчення дії абонементу, кількість занять доступну на даний момент та яка надана була спочатку. Під цієї інформацією є дві кнопки.

Перша кнопка, “Charge”, призначена для поповнення абонемента. Натиснувши на неї відвідувач танцювального центру побачить доступні пакети для поповнення. Вони відрізняються кількістю занять та ціною. Інша кнопка, “Use”, призначена для для того, щоб використати абонемент. Натиснувши на неї, користувачеві відобразиться форма, в якій потрібно буде вказати класи, на які він збирається відвідати. Нижче цих двох кнопок можна побачити список занять, на які було використано цей абонемент. Кожен запис в списку містить інформацію про назву класу та дату його відвідування.

5.3 Опис процесу оплати абонементів

Для відвідування танцювальних занять потрібен абонемент. Зараз в танцювальних центрах абонемент являє собою фізичну картку. Отримати його можна двома способами. Перший полягає в його купівлі у адміністратора танцювального центру. В цьому випадку відвідувач говорить адміністратору свій номер мобільного телефону, який співробітник танцювального центру вводить в систему в застосунку DOM CRM. У випадку якщо це перший абонемент для студента, то адміністратор створює новий абонемент в системі, видає студенту фізичну карту, одразу присвоюючи певний термін

придатності та кількість занять, за яку заплатив студент. Якщо в студента вже був абонемент, то він присвоює оплачені послуги до вже існуючого номера абонемента, не видаючи нової карти.

Другий спосіб оплати абонемента полягає в його купівлі через мобільний застосунок, процес якого візуально зображено в додатку Е. Для цього студент авторизується в мобільному застосунку. Після в боковому меню слід перейти в розділ “Subscriptions”. На сторінці, що відкриється, потрібно натиснути на кнопку “Charge”. Після чого користувач потрапить на сторінку, де потрібно вказати який саме абонемент він хоче придбати.

Доступні види абонементів визначаються менеджером танцювального центру і є різними для кожного бренду. Види абонементів між собою відрізняються трьома критеріями. Перший критерій – це кількість занять, доступних в абонементі. Другий - тип абонемента. В танцювальних центрах вчителі поділені на групи, кожна з яких має свою ціну в залежності від популярності та досвідченості викладачів, які в неї входять. Третім критерієм, який впливає на ціну абонемента є термін його придатності.

Після вибору типу абонемента система виконує запит до платіжної системи. Туди передаються дані, які вимагає платіжна система та ідентифікатор користувача в системі DOM. Після цього користувач перейде на сторінку, яку надає платіжна система. Там йому слід ввести дані кредитної картки. В сучасних платіжних системах, з метою безпеки, користувачеві потрібно зробити ще один додатковий крок. Йому потрібно підтвердити оплату на стороні банку, який випустив кредитну картку. Після цього користувач повернеться в застосунок DOM.

Слід зазначити, набір платіжних систем, які будуть використовуватись для бренду, визначають його менеджери. Інтеграція з платіжними засобами в системі DOM буде відбуватись в компоненті DOM API - серверній частині для мобільного застосунку. Більш детально структуру системи можна побачити в додатку Ж. Цю функціональність не переносять в DOM CRM API, щоб зробити систему безпечнішою та гнучкішою.

Після проведення транзакції платіжна система робить запит на DOM API. В цьому запиті міститься ідентифікатор користувача, по якому визначається якому кому зарахувати оплату абонемента. Після цього виконується авторизований запит, який робить абонемент оплаченим в системі. Послідовність обробки такого запиту можна побачити в додатку И.

Якщо все пройшло успішно, серверна частина мобільного застосунку робить запит за даними про абонемент того ж користувача, щоб відобразити актуальну інформацію в мобільному застосунку. Відповідь цього запиту повертається користувачеві у смартфон.

Також варто вказати, що придбання нового абонемента, замінює вже існуючий, якщо він був. Тобто користувачеві потрібно розуміти, що заняття зі старого абонементу буде втрачено, якщо він придбає новий. Це зроблено з метою звільнення танцювального центру від відповідальності вирішення проблем з абонементами.

5.4 Висновки до розділу

В підрозділі 5.1 описано кожен із застосунків доступних в системі. Вказано призначення кожного з них та принцип їх взаємодії. Також в цьому підрозділі описано принцип використання одним з видів користувачів - адміністраторами системи.

В підрозділі 5.2 описано можливі сценарії використання системи користувачами конкретного бренду - менеджером та адміністратором танцювального центру. Вони взаємодію з DOM CRM, який представлений веб-застосунком. Також цей підрозділ описує мобільний застосунок, який також входить до складу системи. Наведено можливі сценарії його використання вчителями та відвідувачами танцювального центру.

Підрозділ 5.3 описує процес оплати в системі. Покроково описано яким чином оплата відбувається в системі та як проходить взаємодія з платіжною системою.

6 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Нижче наведено маркетинговий аналіз ринку. Це допоможе зрозуміти чи доцільно впроваджувати продукт на ринок. Також такий аналіз дасть відповідь на запитання яким чином краще розвиватись, щоб захопити більшу частину ринку.

6.1 Опис ідеї проекту

Для проведення маркетингового аналізу слід чітко описати ідею стартап проекту. Це дозволить більш точно зрозуміти його переваги та недоліки , що є необіхдним для аналізу. Ідеї стартап-проекту описано в таблиці 6.1. В ній вказано зміст ідеї створення продукту, напрямки його застосування та вигоду, яку отримує користувач від використання продукту. Серед відмінностей по відношенню до рішень-аналогів найбільш вагомими є:

- можливість створювати та керувати розкладом занять. Це в свою чергу робить можливим подальше використання цієї інформації, наприклад, в мобільному застосунку для відвідувачів та вчителів танцювального центру, щоб дізнатися актуальний розклад;
- доступність системи відвідувачам та вчителям танцювального центру. Це можливо за допомогою мобільного застосунку DOM, який також входить до складу системи;
- можливість оплати абонементів онлайн - робить досвід взаємодії з системою простішим, а тому і приємнішим. Також це дозволяє зменшити кількість

Таблиця 6.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди користувача
	Доступ до інформації про розклад занять, список	Полегшує доступ до інформації для всіх

центру. В ній передбачено використання як представниками самої системи, представниками танцювального центру та їх відвідувачами. Вона дозволяє керувати даними про танцювальний центр та його працівників. Також в системі є можливість отримувати статистичну інформацію про ефективність роботи працівників та відстежувати як зміни відобразились на популярності бренду. Система DOM автоматизує певні процеси в роботі танцювального центру або робить інформацію доступною в електронному вигляді. Це значно зменшує кількість контактів між працівниками танцювального центру та його відвідувачами, тим сами знижує ймовірність розповсюдження інфекційних хвороб, які є сьогодні.	викладачів, типи абонементів у танцювального центру та ціни на них в електронному вигляді.	користувачів системи, тим самим економлячи час.
	Створення, видалення та редагування працівників танцювального центру, типів абонементів. Перегляд статистичних даних про танцювальний центр.	Зручність та простота керування танцювальним центром. Можливість зробити зміни в системі миттєво. Наявність статистичної інформації для аналізу.
	Оплата абонементів онлайн в будь-який час доби.	Зручність взаємодії з танцювальним центром для відвідувачів танцювального центру, що в свою чергу підвищує популярність бренду.

Після того, як сформовано основну ідею системи, вказано напрямки її застосування, можливі вигоди користувачів та зазначено чим відрізняється система від

рішень-аналогів, слід порівняти техніко-економічні характеристики ідеї з пропозиціями конкурентів. Це зроблено в таблиці 6.2.

Таблиця 6.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характерис-тики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		DOM	Bitrix 24	Мегаплан	MS Dynamic 365			
1	Створення працівників танцювального центру	Функціонал, який потрібний кінцевим користувачам, наявний	Присутній функціонал. Містить надлишкову інформацію (позиція, відділ)	Присутній функціонал. Містить надлишкову інформацію (позиція, відділ)	Присутній функціонал. Містить надлишкову інформацію (позиція, відділ)		+	
2	Отримання статистичних даних	Доступні дані про ефективність роботи кожного адміністратора та викладача танцювального центру	Доступна більш детальна інформація про роботу адміністраторів танцювальних центрів (логування роботи)	Доступна більш детальна інформація про роботу адміністраторів танцювальних центрів (логування роботи)	Доступна більш детальна інформація про роботу адміністраторів танцювальних центрів (логування роботи)	+		

3	Створення розкладу занять	Є можливість створити розклад проведення занять в танцювальному центрі.	Такої функціональності немає	Такої функціональності немає	Такої функціональності немає			+
4	Наявність застосунків для відвідувачів танцювальних центрів	Доступний мобільний застосунок, в якому є доступ до необхідної інформації та можливість оплатити абонемент.	Застосунку немає, оскільки система призначена для більш загального використання, а не лише для танцювальних центрів.	Застосунку немає, оскільки система призначена для більш загального використання, а не лише для танцювальних центрів.	Застосунку немає, оскільки система призначена для більш загального використання, а не лише для танцювальних центрів.			+
5	Оплата абонементів онлайн	Присутня	Відсутня, оскільки в системі немає застосунку для кінцевих	Відсутня, оскільки в системі немає застосунку для кінцевих	Відсутня, оскільки в системі немає застосунку для кінцевих			+

			користувачів, які оплачують абонементи	користувачів, які оплачують абонементи	користувачів, які оплачують абонементи			
6	Ціна за користування системою	2000 грн/міс	4400 грн/міс	3690 грн/міс	Містить модулі, ціна найдешевшого $20 на місяць + плата за кожного користувача $8			+

Тепер, коли визначено що являє собою система та проведено її порівняння з конкурентами, можна судити про конкурентоспроможність продукту. Його ідея чітка та зрозуміла. В порівняння з рішеннями-аналогами він містить відмінності, які роблять його набагато більш привабливим для споживачів.

6.2 Технологічний аудит CRM-системи для танцювального центру

Можливість реалізувати ідею технічно можна визначити за допомогою її технічного аудиту. Він дасть зрозуміти чи реально імплементувати ідею та яким чином це зробити найкраще. Для цього слід визначити чи є наявні технології для реалізації проекту та чи доступні вони. Це слід зробити для кожної з основних ідей проекту окремо. Також в технічному аудиті наводиться його висновок. Оскільки технологій може бути кілька, слід обрати одну для реалізації кожної з ідей. Аудит наведено в таблиці 6.3.

Таблиця 6.3 — Технологічна здійсність проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Оплата абонементів онлайн	Інтеграція з платіжними системами; Взаємодія з банками напряму;	Є багато платіжних систем; Велика кількість банків;	□ У будь-якої платіжної системи існує документація по взаємодії з нею; □ У банків документацію для взаємодії потрібно запитувати в працівників;
2	Авторизація за допомогою СМС	Інтеграція з сервісом надсилання СМС (Infobip, Vonage, Twilio); Інтеграція з провайдером зв'язку для надсилання СМС повідомлень; JWT; Відкритий токен;	Сервісів надсилання СМС повідомлень є чимало; Наявні бібліотеки на різних мов програмування для формування JWT;	Документація по інтеграції доступна на сайтах сервісів надсилання СМС повідомлень; Принцип роботи JWT описаний та доступний для вільного використання;

3	Авторизація по стандарту OAuth 2.0.	Мови програмування: C# Java JavaScript PHP Python Для передачі корисних даних: JWT дані у звичайному форматі	Мови програмування наявні; Способи передачі інформації наявні;	Всі мови програмування доступні для використання; Способи передачі корисних даних доступні для використання;
4	Автоматична генерація документація для API.	Технології для автоматичної генерації API документації: Swagger; Apiary; Mashape;	Технології наявні	Технології доступні для використання
Для реалізації оплати абонементів онлайн обрано інтеграцію з платіжними системами. Для надсилання повідомлень при авторизація за допомогою СМС обрано інтеграцію з сервісом InfoBip, а для передачи корисних даних використано JWT. Для Імплементації авторизації по стандарту OAuth 2.0 використано мову програмування C#. Автоматична генерація документації для API реалізована з використанням технології Swagger.				

Технологічний аудит показав, що існує кілька технологій для реалізації кожної з ідей продукту. Також є вибір технологій, за допомогою яких це можна зробити. Всі вони доступні для використання.

6.3 Аналіз ринкових можливостей запуску стартап проекту

З метою визначення напрямку розвитку проекту в залежності від стану ринкового середовища слід провести аналіз ринку. Почати варто з аналізу попиту. Його наведено в таблиці 6.4.

Таблиця 6.4 — Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	1
2	Загальний обсяг продаж, грн/ум.од	2.8 млн/рік
3	Динаміка ринку (якісна оцінка)	повільно зростає
4	Наявність обмежень для входу (вказати характер обмежень)	немає
5	Специфічні вимоги до стандартизації та сертифікації	немає
6	Середня норма рентабельності в галузі (або по ринку), %	32%

На даний момент ніша танцювального бізнесу в контексті наявності програмного забезпечення є вільною через низьку прибутковість, проте попит на проект є, про що свідчить загальний обсяг продаж.

Також середня норма рентабельності є вищою, ніж відсоток від банківських вкладень. Тому вхід в цей ринок не є складним, проте існує велика ймовірність, що він є неокупним. Для уникнення побоювань можна зробити цю сферу більш прибутковою шляхом її діджиталізації.

Щоб зрозуміти споживачів, слід їх охарактеризувати. Для цього в таблиці 6.5 їх поділено на кілька груп та проведено аналіз кожної з них. Це зроблено на основі потреб, що формують ринок вцілому. Для кожної з потреб визначено відмінності у поведінці різних цільових груп та підсумовано перелік вимог кожної з таких груп потенційних клієнтів стартап-проекту.

Таблиця 6.5 – Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Створити або налагодити процес занять в танцювальному центрі, зробити його більш прибутковим	Власники танцювальних центрів; Люди, яким делеговано керувати танцювальним центром	Використання різних інтегрованих платіжних систем.	Зручність використання системи; Безпека даних; Зрозуміла та інформативна візуалізація даних;

2	Швидкий доступ до розкладу занять, цін на абонементи	Відвідувачі танцювального центру; Працівники танцювального центру;	Вартість використання продукту та додаткового обладнання, точність підрахунку, вплив на відвідуваність закладу	Зручний формат подачі інформації; Доступ з мобільних пристроїв;
3	Можливість придбати абонемент онлайн	Відвідувачі танцювальних центрів	Різна комісія за сплату через використання різних платіжних систем	Безпека даних; Простота виконання;

В ході аналізу для кожної з потреб, що формує ринок визначено групи користувачів. Також вказано вимоги цих груп до проекту та відмінності у їх поведінці.

Після формування опису кінцевих споживачів потрібно визначити фактори які позитивно та негативно впливають на впровадження продукту на ринок. Для цього сформовано дві таблиці. В таблиці 6.6 наведено перелік загроз, які є на ринку та можуть зашкодити успішному запуску.

Таблиця 6.6 — фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Непопулярність сфери танців	Танцювальний бізнес не є популярним сьогодні. Хоча існує багато танцюристів в нашій країні, проте вони не мають змоги	Проведення рекламної кампаній двох напрямків:

		відвідувати танцювальні центри, щоб розвиватись.	показати ймовірним власникам попит на танцювальні центри; показати танцюристам спосіб розвиватися в танцювальних школах;
2	Низькі прибутки	Через непопулярність сфери танців прибутки є низькими	Популяризація сфери танців шляхом реклами
3	Пандемія в світі	Через негативну статистику захворюваності в країні можливий повторний локдаун	Додавання функціоналу проведення занять онлайн. Зміна формату на платформу з відео танцювальних занять.

Наведено кілька факторів загроз, проте на кожну з них у компанії продумано як реагувати. Таблиця 6.7 містить опис факторів, які надають можливості для прискореного росту продукту після його впровадження.

Таблиця 6.7 — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Відсутність прямих конкурентів	В сфері розробки програмного забезпечення для танцювального бізнесу немає конкурентів, які заточені виключно під неї	Швидке захоплення існуючих танцювальних центрів через рекламу

2	Популярність мобільних пристроїв	Сьогоденне суспільство дуже часто використовує мобільні пристрої в повсякденному житті	Створення застосунків для мобільних пристроїв для підвищення зручності використання користувачів
3	Введення повторного карантину	Введення повторного карантину змусить танцювальні центри перейти в онлайн режим роботи	Створення функціональності для проведення занять онлайн

З переліку наведених можливостей найбільше значення має відсутність прямих конкурентів. Це дасть змогу скоріше зайти на ринок та отримати велику аудиторію меншою ціною.

Проте є рішення, які вже використовуються танцювальними центрами. Хоча вони не заточені виключно під цю сферу, вони є непрямими конкурентами, з якими потрібно боротись. Загальні риси конкуренції, які присутні в сефері танцювального бізнесу наведено в таблиці 6.8.

Таблиця 6.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - монополія	Існує одна CRM-система, яку використовує тотальна більшість танцювальних центрів	Створення CRM-системи для виключно сфери танцювального бізнесу

2. За рівнем конкурентної боротьби - національний	Створена система передбачається використовуватись виключно у національних масштабах	Заохочувати танцювальні центри давати заняття в різних містах
3. За галузевою ознакою - внутрішньогалузева	Система призначена для користувачів виключно однієї галузі	Створення більш специфічної, вузько направленої функціональності
4. Конкуренція за видами товарів: - товарно-родова	Функціональність продуктів схожа лише частково через загальну призначеність продуктів конкурентів	Використати досвід використання наявного у конкурента потрібного функціоналу та доповнити
5. За характером конкурентних переваг - нецінова	Потенційні споживачі системи звертають увагу на наявність більш профільного функціоналу	Забезпечення функціоналу необхідного споживачам
6. За інтенсивністю - марочна	Для клієнтів важливе значення має бренд	Розвиток бренду

Для більш детального аналізу додатково конкуренцію в цій сфері розглянуто за моделлю М. Портера. Його наведено в таблиці 6.9.

Таблиця 6.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Bitrix 24; Мегаплан;	тематична функціональність і зручність використання;	Постачальників немає	зручність використання; доступність даних;	Направленість системи на певну тематику
Висновки	Bitrix 24 - прямий конкурент - найпоширеніша система в цій галузі, проте вона не містить всієї функціональності для танцювальної сфери. Мегаплан - не прямий конкурент, оскільки він не має всієї функціональності для танцювальної сфери та не	Умови виходу на ринок сприятливі, оскільки існує лише один прямий конкурент, який не є орієнтований на саме цю сферу. Термін, за який планується вийти на ринок,	В сфері танцювального бізнесу немає постачальників, які диктують умови.	Клієнти визначають потреби систем цієї сфери	Продукти-замінники створюють певні складнощі, адже користувачі вже звикли до них

	поширений в ній. Конкуренція я не інтенсивною через те, що конкуренти не сфокусовані виключно на цій сфері	становить 1 рік.			

Проаналізувавши таблицю 6.9 можна зробити висновок, що на ринку присутній лише один прямий конкурент - Bitrix 24. Інші CRM-системи є не популярними серед цільової аудиторії, оскільки призначенні більше для продаж. Також ціна таких систем є вищою ніж в Bitrix 24. Тому їх не слід вважати прямими конкурентами. Оскільки навіть прямий конкурент є не надто поширеним в цій сфері в силу відсутності важливих функцій в нього, можна вважати ринок відкритим для заповнення. Тому оцінка виходу на ринок є досить оптимістичною та становить 1 рік.

Враховуючи аналіз конкурентності на ринку та вимоги споживачів до товару можна визначити фактори по яким продукт є конкурентоспроможний. Їх наведено та обгрунтовано в таблиці 6.10.

Таблиця 6.10 — Обгрунтування факторів конкуроентоспроможності

№	Фактор конкурентоспромо жності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Більше функціональності	У рішень-аналогів немає можливості налаштовувати розклад занять в системі

2	Доступність різним типам користувачів	Оскільки в конкурентів немає можливості керувати розкладом занять в танцювальному центрі, то немає інформації, яку можна подавати відвідувачам танцювальних центрі. У випадку створеної системи, така інформація є і доступна клієнтам бренду в мобільних застосунках
3	Зручність використання	Зручний інтерфейс є і в рішеннях аналогів. Проте не у всіх є мобільні застосунки в системі. Саме з мобільних пристрій доступ до системи найзручніший та найшвидший
4	Оплата абонементів онлайн	Такої функціональності немає в системах конкурентів, а вона є дуже корисною та актуальною в умовах епідеміологічної ситуації, яка зараз в світі

Після того, як визначено які фактори грають на користь продукту, слід зробити порівняльний аналіз сильних та слабких його сторін. Такий аналіз наведено в таблиці 6.11.

Таблиця 6.11 — Порівняльний аналіз сильних та слабких сторін системи

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з агрегатором						
			–3	–2	–1	0	+1	+2	+3
1	Більше функціональності	20				+			
2	Доступність різним типам користувачів	17	+						

3	Зручність використання	15	+						
4	Оплата абонементів онлайн	15	+						

Результатом ринкового аналізу можливостей продукту є формування матриці аналізу сильних та слабких сторін, загроз та можливостей. Її наведено в таблиці 6.12.

Таблиця 6.12 — SWOT-аналіз стартап-проекту

Сильні сторони: Більше функціональності, доступність різним типам користувачів, зручність використання, оплата абонементів онлайн	Слабкі сторони: Менша кількість статистичних даних, невідомість бренду
Можливості: збільшений попит, використання можливостей мобільних пристроїв	Загрози: знижений інтерес потенційних клієнтів використовувати систему, складність захоплення ринку

Виходячи з даних, отриманих в результаті аналізу слабких та сильних сторін стартап-проекту, можна сформувати альтернативи впровадження стартап-проекту на ринок. Для цього необхідно описати альтернативи, вказати ймовірність отримання необхідних ресурсів та строк реалізації кожної з них. Перелік альтернатив для CRM-системи танцювального центру наведено в таблиці 6.13. Слід зазначити, що термін реалізації вказаний при невеликій команди розробки в кількості трьох людей.

Таблиця 6.13 — Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Використання NFC смартфонів, як заміни пластикових карток-абонементів	80%	3 місяця
2	Впровадження онлайн платформи для танцювальних занять	У випадку: ускладнення епідеміологічної ситуації - 95%; полегшення епідеміологічної ситуації - 35%	9 місяців

Після формування альтернатив можна зробити висновок, що доцільніше реалізувати використання NFC смартфонів для заміни пластикових карток-абонементів. Оскільки ймовірність отримання ресурсів у неї досить велика, а строк реалізації в три рази менший, ніж в іншої альтернативи.

6.3 Розробка ринкової стратегії проекту

Ринкова стратегія проекту допоможе визначити цільові групи потенційних користувачів. Їх наведено в таблиці 6.14. В ній міститься опис цільової групи, її

готовність споживати продукт, попит на продукт , інтенсивність конкуренції на ринку та простоту входу у сегмент.

Таблиця 6.14 — Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Танцювальні центри, що використовують одну з існуючих CRM	Система створена легша у використанні, ніж аналоги, тому - готові	В системі можливо керувати розкладом занять, що є дуже важливим для танцювальних центрів, тому - вище середнього	Оскільки для конкурентів сфера танцювального бізнесу не є цільовою, тому - нижче середньої	Через наявність можливості керувати розкладом, переконати клієнтів буде відносно легко, тому - нижче середньої
2	Танцювальні центри, які не використовували CRM	Система створена виключно для цієї сфери, тому всі поняття в ній знайомі	Епідеміологічне становище підвищує потребу використання, тому - високий	Через відсутність конкурентів, які захоплюють решту частину ринку,	Через відсутність конкурентів і високий попит - дуже просто

		споживачам. Це полегшить початок користування системою, тому - готові		інтенсивність конкуренції - низька	
Які цільові групи обрано: 1, 2					

З результатів аналізу, можна зрозуміти, що для розвитку проекту буде використано стратегію розвитку, оскільки обрано одразу кілька цільових груп. Ці дані використано для формування базової стратегії розвитку. Її наведено в таблиці 6.15, в якій відображено обрану альтернативу розвитку проекту, стратегію охоплення ринку, ключові конкурентоспроможні позиції відповідно до обраноїальтернативи та базову стратегію розвитку.

Таблиця 6.15 Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Впровадження онлайн платформи для танцювальних занять	Масовий маркетинг	Можливість проведення занять в онлайн форматі дає можливість переглядати записи в зручний користувачеві час	Стратегія диференціації

Щоб вирішити, як компанії вести себе відносно конкурентів, визначено базові стратегії конкурентної поведінки. Їх наведено в таблиці 6.16.

Таблиця 6.16 — Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Так, оскільки конкуренти створені для багатьох сфер	Так	Ні	Наступальна стратегія лідера

З обраних вимог споживачів та вибраної базової стратегії розвитку та стратегії конкурентної поведінки визначено стратегію позиціонування. Вона наведена в таблиці 6.17 і полягає у формуванні позицій за якими споживачі будуть ідентифікувати бренд.

Таблиця 6.17 — Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
	Наявність необхідного функціоналу, зручність використання	Стратегія диференціації	Можливість керувати розкладом занять та його доступ відвідувачам	Орієнтованість виключно на сферу танцювальних центрів, зручність не лише для працівників

			танцювального центру	танцювальних центрів, а й для їх відвідувачів

В результаті визначення стратегії позиціонування, можна зробити висновок, що компанія буде забезпечувати користувачів усім необхідним функціоналом для танцювальних центрів та надасть необхідну зручність використання. Така позиція компанії разом з стратегією диференціації та наступальною стратегією лідера забезпечать успіх при впровадженні системи на ринок.

6.4 Розроблення маркетингової програми стартап-проекту

Важливим є правильно подати продукт користувачу. Для цього розроблено маркетингову програму стартап-проекту. Вона починається з переліку ключових переваг концепції потенційного товару, які наведено в таблиці 6.18. Там міститься потреби до продукту, вигоди, які він пропонує та переваги перед конкурентами.

Таблиця 6.18. Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Доступність даних в електронному вигляді	Користувачі мають доступ до відповідних даних: менеджери та адміністратори танцювальних центрів мають доступ до CRM-системи. Вчителі та відвідувачі танцювальних центрів через	Наявність застосунків для відвідувачів танцювальних центрів

		мобільні застосунки можуть побачити розклад занять	
2	Оплата абонементів онлайн	Можливість придбати абонемент через мобільний застосунок	Оскільки у конкурентів немає такої функціональності, її присутність в створеній системі вже є перевагою
3	Можливість керувати розкладом занять	Можливість миттєво вносити зміни в розклад занять	У конкурентів немає можливості керувати розкладом занять в танцювальному центрі

Далі, в таблиці 6.19 наведено трирівневу модель товару, в якій вказано яким чином товар буде доступний споживачеві.

Таблиця 6.19 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	CRM-система для танцювальних центрів		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/О р
	Властивості: До складу системи входить веб-застосунок для керування танцювальним центром та мобільний застосунок для відвідувачів танцювального центру	НМ	ТХ/ТЛ

	Якість: актуальність функціоналу, доступність та зручність використання системи
	Пакування: немає
	Марка:
	До продажу: базова версія
	Після продажу: додавання нових можливостей, постійне покращення продукту
За рахунок чого потенційний товар буде захищено від копіювання: захист ідеї товару(захист інтелектуальної власності)	

Визначено межі цін на продукт. Їх наведено в таблиці 6.20. Їх сформовано на основі цін на товари аналогів та середній рівень доходів цільової групи користувачів продукту.

Таблиця 6.20 — Визначення меж встановлення цін

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	Немає	Оскільки в систем-аналогів йде додаткова плата за користувачів в системі, то немає можливості точно	Близько 60 тис. грн/місяць	3500 - 6000 грн/місяць

		вказати ціну за використання. Близько 2500 - близько 5000 грн/місяць		

В таблиці 6.21 описано, яким чином буде надана можливість використання системи споживачу.

Таблиця 6.21 — Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Наявність необхідного функціоналу та доступність системи	Створити обліковий запис бренду в системі та додати менеджерів танцювального центру	2	Через рекламну кампанію

На основі раніше обраного позиціонування системи сформовано сформовано концепції маркетингових комунікацій. Їх наведено в таблиці 6.22. Визначено завдання рекламного повідомлення та концепцію рекламного повідомлення.

Таблиця 6.22 — Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення

	Цільові клієнти керують танцювальними центрами вручну або за допомогою систем аналогів, в яких немає важливих функцій, що є в створній системі	Месенджери та соціальні мережі. В рідких випадках - телефонний зв’язок	можливість керувати розкладом занять; доступність системи різним користувачам можливість оплати онлайн	Показати функціонал, якого не вистачає в системах аналогах	Оскільки власники танцювальних центрів та відвідувачі часто використовують соціальні мережі, тому - SMM

6.5 Висновки до розділу

В розділі 6 проведено маркетинговий аналіз ринку. За ним визначено, що можливість ринкової комерціалізації проекту та перспективи впровадження потенційним групам клієнтів присутні. Сформовано перелік альтернатив впровадження системи та обрано найбільш оптимальну з них. Як результат, можна сказати, що подальша імплементація проекту є доцільною.

ВИСНОВКИ

Під час виконання магістерської дисертації було проведено огляд предметного середовища та проаналізовано існуючі CRM-системи танцювальних центрів. На основі проведеного аналізу сформовано порівняльну таблицю рішень-аналогів та визначено їхні переваги та недоліки. Також в результаті вищезгаданого аналізу було сформовано проблеми, які повинна вирішувати система та наведено перелік вимог до системи. Їх поділено на функціональні та технічні.

Після формування вимог було спроектовано архітектуру системи та обрано підхід, який покладено в основу роботи проекту. Відштовхуючись від цього було обрано набір технологій, який використовувався для реалізації системи. Кожну технологію було описано та доведено її переваги перед аналогами. Створену архітектуру, підхід та обрані технології використано при розробці системи.

Результатом реалізації є система, що складається з двох застосунків, призначених для різних типів користувачів. Кожен з них має клієнт-серверну архітектуру. Серверні частини застосунків написані з використанням техтології ASP.NET Core, що дає змогу розгортати їх на будь-яких сучасних операційних системах. Один клієнтський застосунок реалізований з використанням фреймворку React і представляє собою веб-застосунок. Інший – мобільний застосунок, виконано з використанням технології Xamarin Forms. Дані в системі зберігаються в реляційній СУБД MSSQL.

В записці наведено процес реалізації різних компонентів системи, обгрунтовано рішення та піходи, які були прийняті під час розробки. Для ключових аспектів системи наведено більш детальну інформацію в діаграмах послідовностей, розгортання, та взаємодії між класами. Описано різні способи авторизації, які використовуються в системі. Візуальне відображення принципу роботи кожного з них наведено в додатках.

Було детально описано сценарій використання системи. Наведено загальний принцип роботи системи, а також розглянуто сценарії використання кожного із

застосунків, наявних в системі. Візуально цю інформацію зображено в use-case діаграмі. Також вказано яким чином буде відбуватись оплата абонементів танцювального центру.

Також було доведено екномічну доцільність створення системи як стартап-проекту. Проведено технологічний аудит CRM-системи танцювального центру для визначення можливостей реалізації, проаналізовано ринкові можливості для запуску проекту, щоб зрозуміти потенціал ринку. Також розроблено стратегію розвитку та маркетингову програму.

Як результат створено систему, яка містить застосунки для парцівників танцювальних центрів та їх відвідувачів. Відрізняється від конкурентів наявністю більшої функціональності, необхідної для танцювальних центрів та доступністю необхідної інформації в електронному вигляді.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бітрікс24: Вартість CRM системи хмарне рішення в Бітрікс24 [Електронний ресурс] Ціна на CPM систему Бітрікс24 хмарне рішення - bitrix24.ua Назва з екрана. Доступ : https://www.bitrix24.ua/prices/

2. Цена CRM системы - купить CRM Мегаплан, стоимость тарифов системы [Електронний ресурс] Сколько стоит CRM-система - зависит от функциональных возможностей. Выберите среди наших тарифов подходящий вам и купите CRM в несколько кликов! Назва з екрана. Доступ : https://megaplan.ru/calculation/

3. Pricing | Microsoft Dynamics 365 [Електронний ресурс] Find Microsoft Dynamics 365 plans and pricing to choose the applications that are right for your business needs. Назва з екрана. Доступ : https://dynamics.microsoft.com/en-us/pricing/

4. Cryptography Stack Exchange [Електронний ресурс] : Authentication - SHA1 usage for passwords, alternatives and advantages. Назва з екрана. Доступ : https://crypto.stackexchange.com/questions/624/sha1-usage-for-passwords-alternatives-and-advantages

5. Devcolibri [Електронний ресурс] : Что такое ООП и с чем его едят? Назва з екрана. Доступ : https://devcolibri.com/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-%D0%BE%D0%BE%D0%BF-%D0%B8-%D1%81-%D1%87%D0%B5%D0%BC-%D0%B5%D0%B3%D0%BE-%D0%B5%D0%B4%D1%8F%D1%82/

6. CQRS [Електронний ресурс] CQRS (Command Query Responsibility Segregation) is the notion that you can use a different model to update information than the model you use to read information Назва з екрана. Доступ : https://martinfowler.com/bliki/CQRS.html#:~:text=CQRS%20stands%20for%20Command%20Query,you%20use%20to%20read%20information.

7. Введение в CQRS + Event Sourcing: Часть 1. Основы [Електронний ресурс] : CQRS расшифровывается как Command Query Responsibility Segregation. Это паттерн проектирования, о котором я впервые услышал от Грега Янга. В его основе лежит простое понятие, что вы можете использовать разные модели для обновления и чтения информации. Назва з екрана. Доступ : https://habr.com/ru/post/146429/

8. The Model-View-ViewModel Pattern – Xamarin [Електронний ресурс] : This chapter explains how the eShopOnContainers mobile app uses the MVVM pattern to cleanly separate the business and presentation logic of the app from its user interface. Назва з екрана. Доступ : https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

9. An Introduction to OAuth 2 | DigitalOcean [Електронний ресурс] : OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Facebook, GitHub, and DigitalOcean. It works by delegating user authentication to the service that hosts the user acc. Назва з екрана. Доступ : https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

10.Полторак В.П. Криптографічний захист даних в цифрових інформаційних системах (Частина 3) / Полторак В.П. // Телеком. Військовий зв’язок. Спеціальний випуск №2, 2019. - К.: Softpress. hi-Tech.ua, Жовтень, 2019. - с. 64-67. Мова публікації: українська.

11.Полторак В.П. Криптографічний захист даних в цифрових інформаційних системах (Частина 2) / Полторак В.П. // Телеком. Військовий зв’язок. Спеціальний випуск №1, 2019. - К.: Softpress. hi-Tech.ua, Квітень, 2019. - с. 81-85. Мова публікації: українська.

12.Полторак В.П. Криптографічний захист даних в цифрових інформаційних системах (Частина 1) / Полторак В.П. // Телеком. Військовий зв’язок. Спеціальний випуск №2, 2018. - К.: Softpress. hi-Tech.ua, Жовтень, 2018. - с. 98-104. Мова публікації: українська.

13.RFC 7519 - JSON Web Token (JWT) [Електронний ресурс] : JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted. Назва з екрана. Доступ : https://tools.ietf.org/html/rfc7519

14.SASS vs LESS vs SCSS - That are the differences – IONOS [Електронний ресурс] : SASS adds features to CSS and makes work easier. We explain what’s behind the language and the differences between the SASS and LESS preprocessors. Назва з екрана. Доступ: ttps://www.ionos.com/digitalguide/websites/web-development/sass/

15.React vs Vue: What is the best choice for 2020? : Web and Mobile Development Blog – MindK.com [Електронний ресурс] : We compared Vue vs React in terms of performance, scalability, ease of deployment, ecosystem and talent pool. Click to find out which JS tech won the battle. Назва з екрана. Доступ : https://www.mindk.com/blog/react-vs-vue/